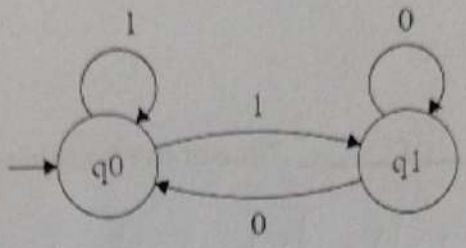
Tutorial #	1	Student ID	2300032601
Date		Student Name	M. KOUSHIK

Pre-Tutorial (To be completed by student before coming to tutorial class)

1. Is the following transition diagram a DFA? Explain.



Solution:

No, the following is not a DFA

Because

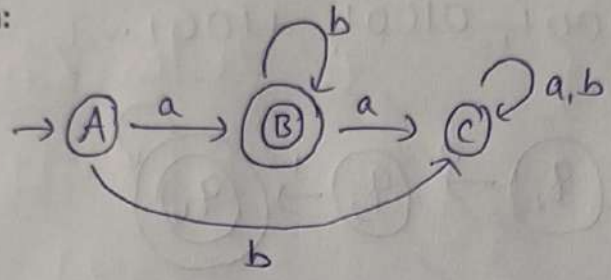
1) There is no final state in the diagram
2) q_0 has 2 transition on 1 & has no transition on '0'
3) q_1 has 2 transition on 0 & has no transition on '1'

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. What is the language recognized by the following DFA? The initial state is 'A' and final or accepting state is 'B'. Draw the DFA and explain learner answer.

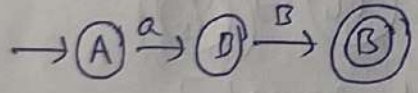
State	a	b
A	B	C
B	C	B
C	C	C

Solution:



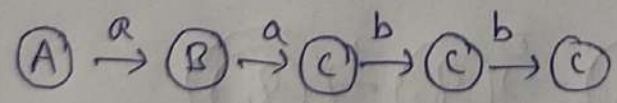
L = {w/w contains only one 'a' starting followed by any number of b's}

String ab



B is a final state, it is accepted

solving aabb



It is not final state, so it is Rejected

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

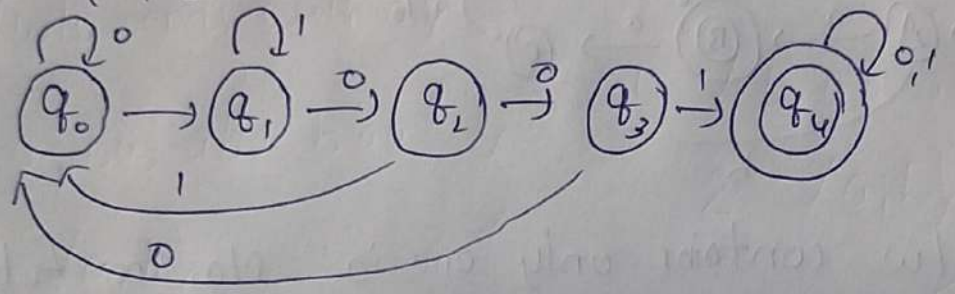
3. Construct a DFA that accepts the language

$$L = \{w \in \{0,1\}^* : w \text{ contains } 1001\}$$

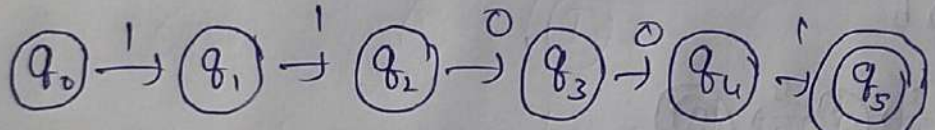
Solution:

$L = \{w / w$ 1001 has sub string$\}$

$\Sigma = \{0, 1\}$ $L = \{1001, 01001, 11001 \ldots\}$

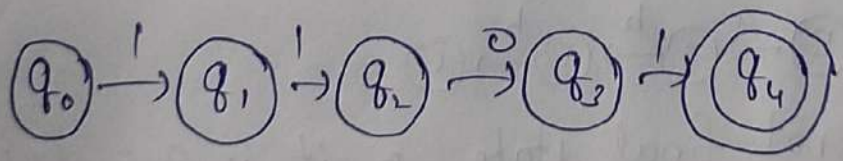


String 11001



q_5 is final state so, it is accepted

string 1101

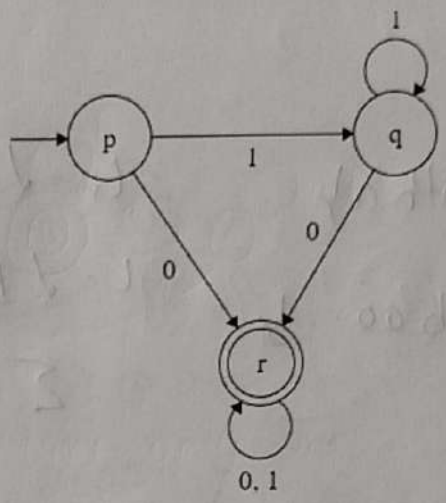


q_0 is not final state so, it is Rejected

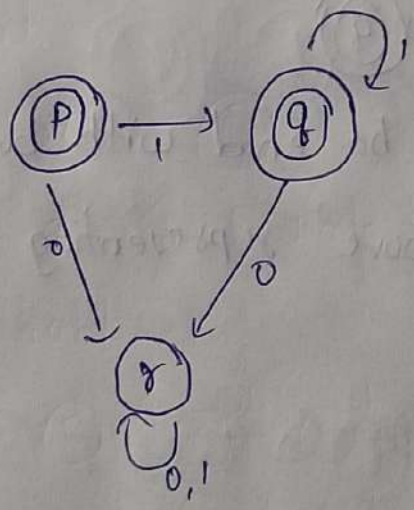
Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out by student in presence of faculty in classroom)

1. Find the complement of the language accepted by the following DFA:



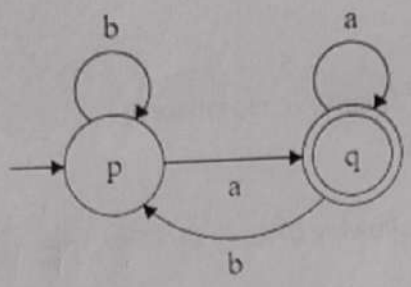
Solution:



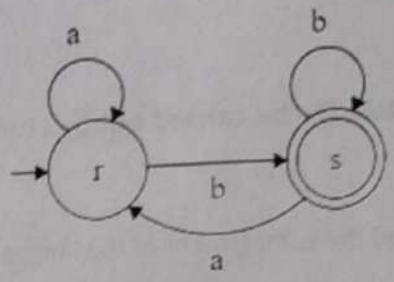
Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Consider the DFAs M and N given below. Construct a DFA that accepts the langauge $L(M) \cap L(N)$.

M:



N:



Solution:

$L_1 = \{w | w \text{ ends with } a\}$

$L = \{a, ba, aba, baa \cdots\}$

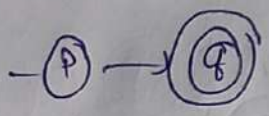
$\Sigma = \{a, b\}$

$L_2 = \{w | w \text{ ends with } b\}$

$L = \{b, ab, abb, aabb \cdots\}$

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

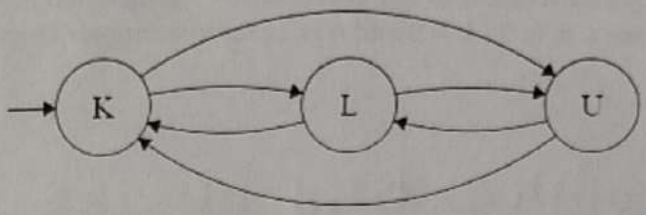
$L_1 \cap L_2 = \emptyset$



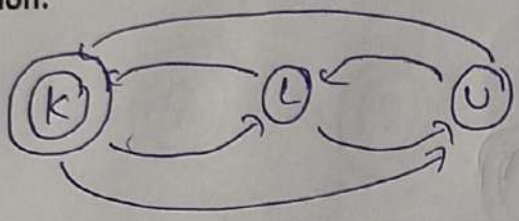
Here, No string can be end with both 'a' & 'b'

Sort the DFA is above representing as $\emptyset$

3. Consider the following DFA that accepts the language L. Draw a DFA that is obtained by reversal of the given DFA.



Solution:



L = { 110, 1000, 0010 ... }

L = { 011, 0001, 0100 ... }

String 0001

U →0 L →0 K →0 U →1 K

K is a final state so, it is accepted

String 1101

U →1 K →1 L →0 K →1 L

L is not final state so, it is rejected

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

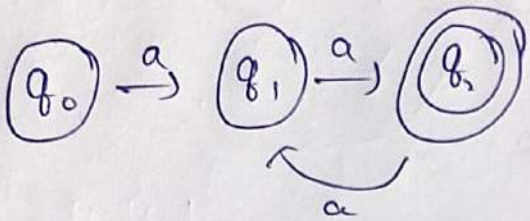
4. Design a DFA for the language $L = \{a^{nk} | \, k > 0 \text{ and } n \text{ is a positive integer constant}\}$ for $\Sigma = \{a\}$.

Solution:

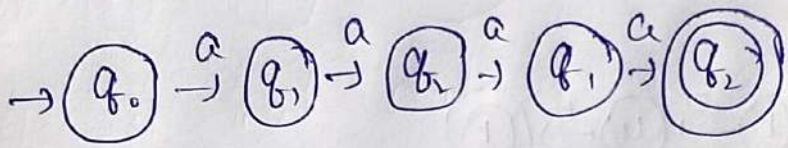
$n = 2$

$L = \{a^2, a^4, a^6 \ldots\}$

$\Sigma = \{a\}$

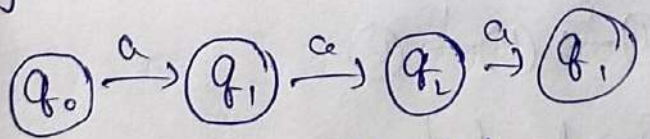


string - a^4



q_2 is final state so, it is accepted

string - a^3



q_1 is not final state so, it is rejected

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

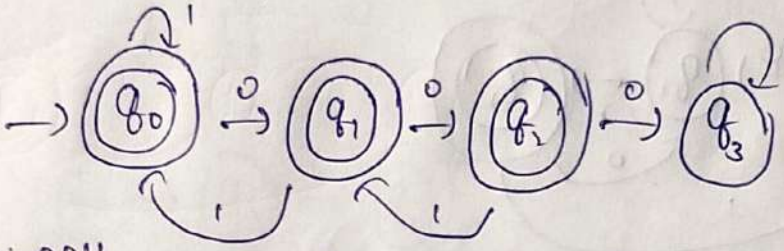
5. Design a DFA for the following:

Consider the set of strings on {0,1} in which, every substring of 3 symbols has at most two zeros. For example, 001110 and 011001 are in the language, but 100010 is not. All strings of length less than 3 are also in the language.

Solution:

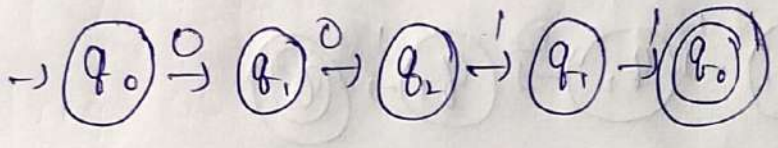
$L = \{001110, 011001, 101 \ldots\}$

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

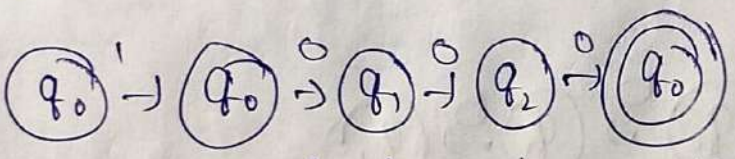


String:-0011

q_0 is final state so, it is accepted



String:- 1000



q_0 is not final state so, it is rejected

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

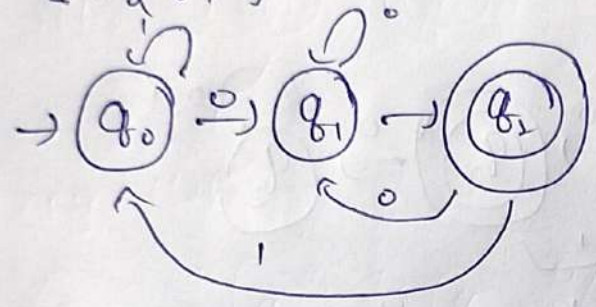
Post-Tutorial (To be carried out by student after tutorial session is over)

1. Draw a DFA for the language accepting strings ending with '01' over input alphabets ∑ = {0, 1}

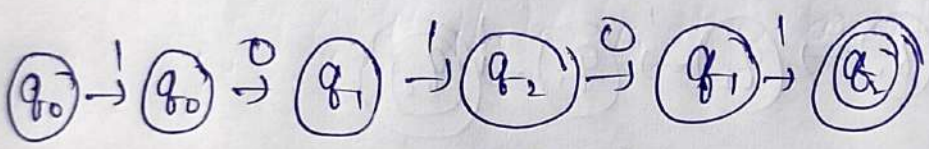
Solution:

L = { 01, 001, 1001, 0101 ... }

Σ = { 0, 1 }

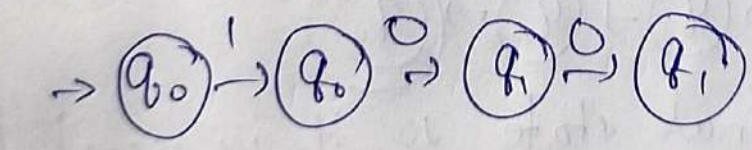


String 10101



q_2 is final state it is accepted

String 100



q_1 is not final state so rejected

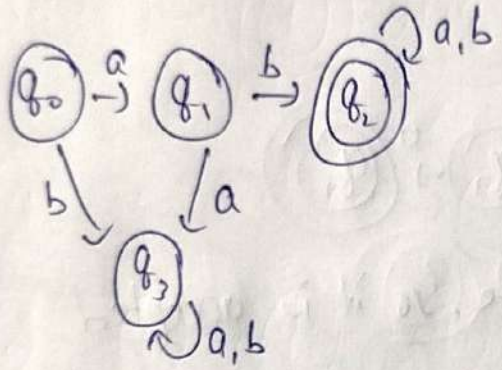
Tutorial #		Student ID	
Date		Student Name	

2.Draw a DFA for the language accepting strings starting with 'ab' over input alphabets Σ = {a, b}

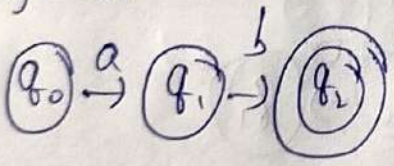
Solution:

$L = \{ab, aba, abb, abab \cdots\}$

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

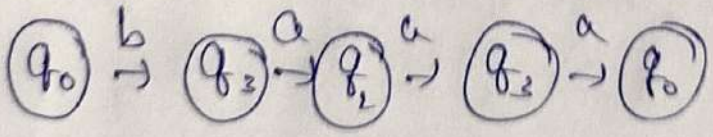


string : ab



q_2 is final state it is accepted

string : baaa



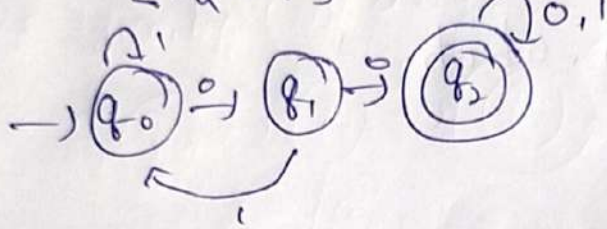
It is rejected

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

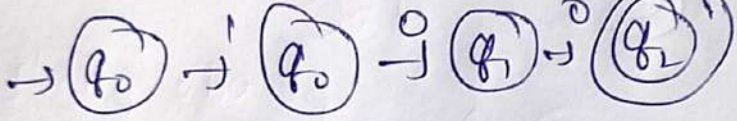
3.Draw a DFA for the language accepting strings having 2 consecutive 0's over input alphabets $\Sigma = \{0, 1\}$.
Ex: 0011, 1001, 1100, 111001100100, and 100001 are all accepted strings.

$L = \{00, 0011, 1001, 1100 \ldots\}$

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

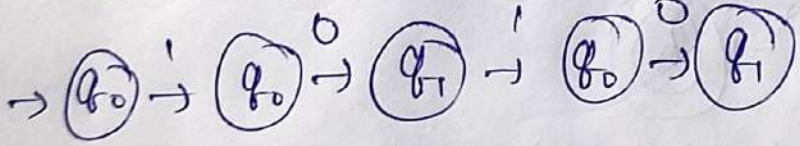


String 100



q_2 is final state, so it is accepted

string = 1010



q_2 is not final state so it is rejected

Tutorial #		Student ID	
Date		Student Name	

Viva Questions:

1. What is the significance of the initial state and final state in a DFA? How do these states impact the acceptance or rejection of input strings by the DFA? Provide examples to illustrate your answer.

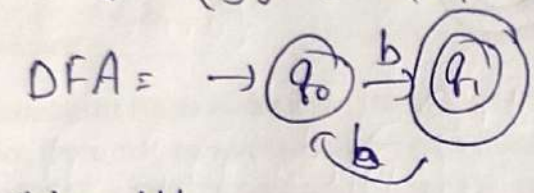
Solution:

The starting state in the DFA is intial state
Destination in the DFA is final state

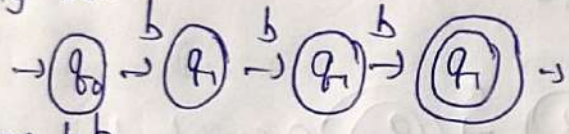
Ex:- $L = \{w | w$ contains 000 No of b's$\}$

$\Sigma = \{b\}$ $L = \{b, bbb, \ldots\}$

DFA =

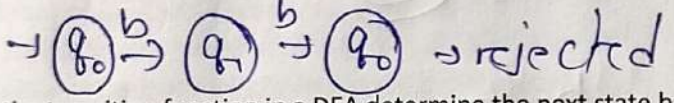


string bbb



→ accepted

string bb



→ rejected

2. How does the transition function in a DFA determine the next state based on the current state and input symbol? Provide a detailed explanation of the transition function and its role in the DFA.

Solution:

'λ' is represented by 5-types

$M = (Q, \Sigma, q_0, \delta, F)$

Where Q = is finite set of state

Σ = is finite input of alphabet

q_0 = is a finite state, $q_0 \in Q$

F = is a set of finite stat, $F \subseteq Q$

$\delta = Q \times \Sigma \to Q$

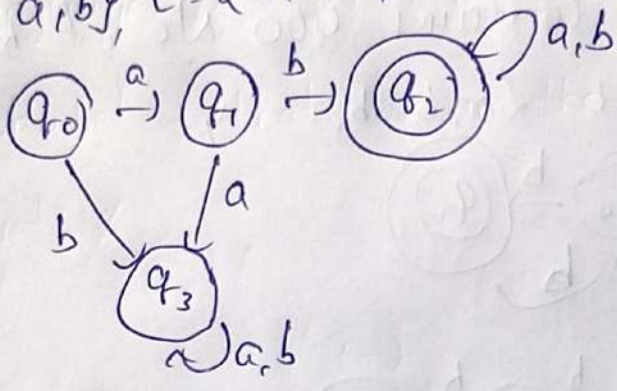
Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Explain the concept of a DFA's language acceptance. How does the DFA determine whether an input string is accepted or rejected? Provide a detailed explanation of the acceptance process.

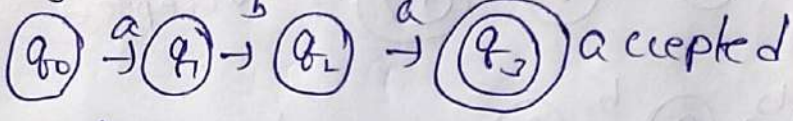
Solution:

Example of DFA: Construct DFA for b-{w/w starts with abb}

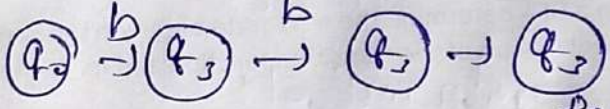
$\Sigma = \{a, b\}$, $L = \{aba, abb, abba \ldots\}$



String aba



String bba



Rejected

Comment of the Evaluator (if Any)	**Evaluator's Observation** Marks Secured:_____out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

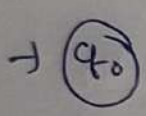
Tutorial #	2	Student ID	2300032601
Date	<TO BE FILLED BY STUDENT>	Student Name	M. KOUSHIK

Pre-Tutorial (To be completed by student before coming to tutorial class)

1. Draw a NFA that accepts the language $\emptyset^*$. Explain your answer.

Solution:

The language $\emptyset^*$ represent the set all string over the alphabet Σ that are in the Klene star of the empty set $\emptyset$



$S = \{q_0\}$

$A = \Sigma$

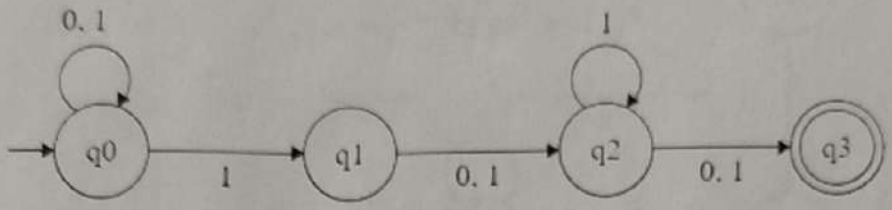
$IS = q_0$

$FS = q_1$

No Transition

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Consider the finite automaton in the following figure.



What is the set of reachable states for the input string 0011? What is the language accepted by the above automaton.

Solution:

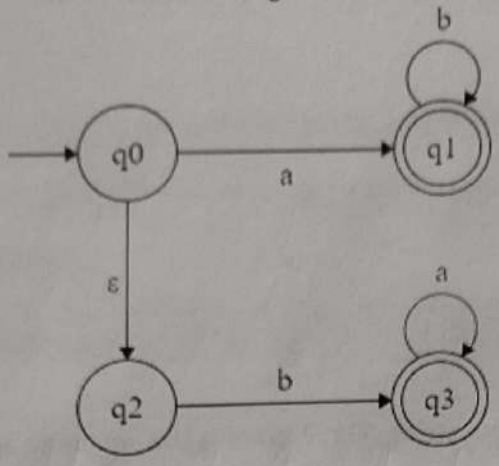
$L = \{(0+1)^* 1 (0+1)^* (0+1)\}$

$L = \{\omega \in \{0,1\}$ condition on odd numbers of 1's$\}$

	0	1
q_0	q_{01}	$q_1 q_2$
q_1	q_2	q_2
q_2	q_3	$q_1 q_3$
$*q_3$	ϕ	ϕ

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Convert the following ε-NFA to NFA without ε- transition.



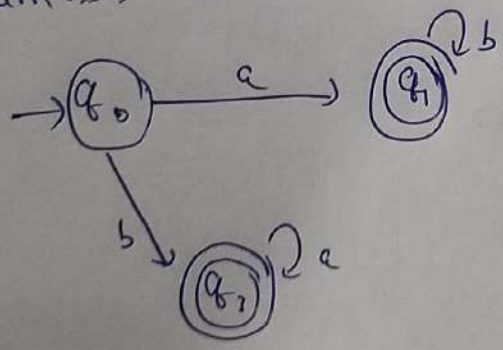
Solution:

ε-NFA to NFA

NFA →

	a	b
q_0	q_1	q_3
q_1*	∅	q_1
q_2*	q_3	∅

⇒ Diagram ....

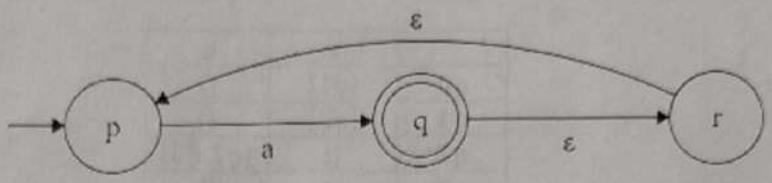


→ If we can read to final state from that node by using 'ϵ' then it will be a final state

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. What is the complement of the language accepted by the NFA shown below? Draw the NFA for the language found out. Assume $\sum = \{a\}$ and ε is the empty string.



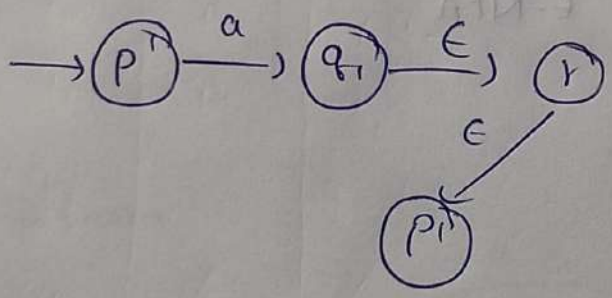
Solution:

State = {p, q, r}

ϵ closure (p) = {p, q, r}

ϵ closure (q) = {q, r}

ϵ closure (r) = {p, r}



Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Consider the following ε-NFA, $M = (\{q0, q1, q2, q3\}, \{a\}, \delta, q0, \{q3\})$ where δ is given in the transition table below. What is the minimum length of string to reach to the final state.

	ε	a
q0	{q1}	{}
q1	{q2}	{}
q2	{}	{q2, q3}
q3	{}	{}

Solution:

$M = (\{q_0, q_1, q_2, q_3\}, \{a\}, \{q_0, q_3\})$

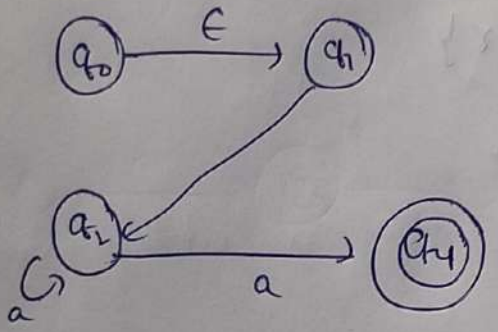
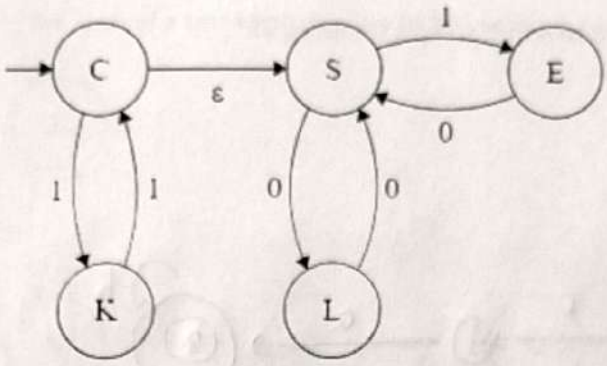


Diagram of ϵ-NFA

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Convert the following ε-NFA to NFA without ε-transitions.

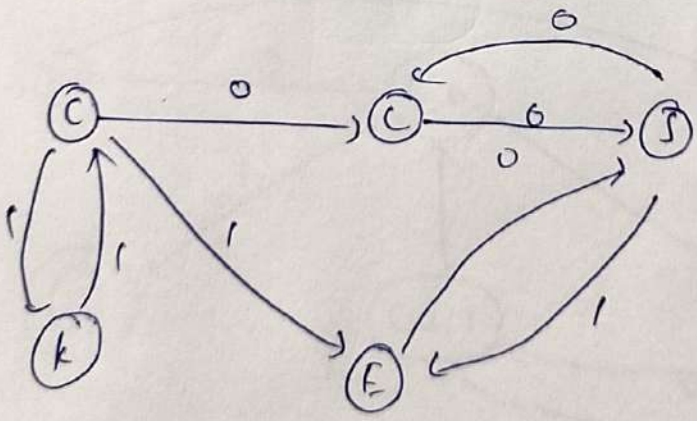


Solution:

Table of NFA

	0	1
C	L	k, E
L	S	∅
k	∅	C
E	S	–
S	L	E

Diagram



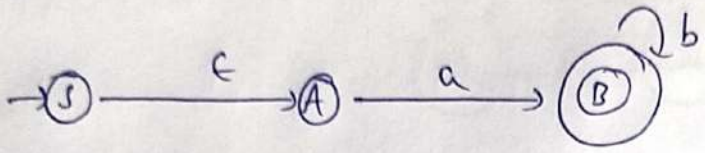
Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after Tutorial session is over)

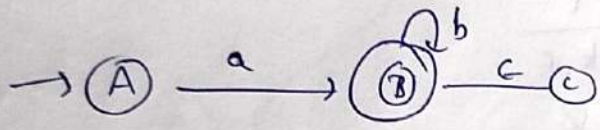
1. Construct an ε-NFA that accepts the string ab*.

Solution:

ab*

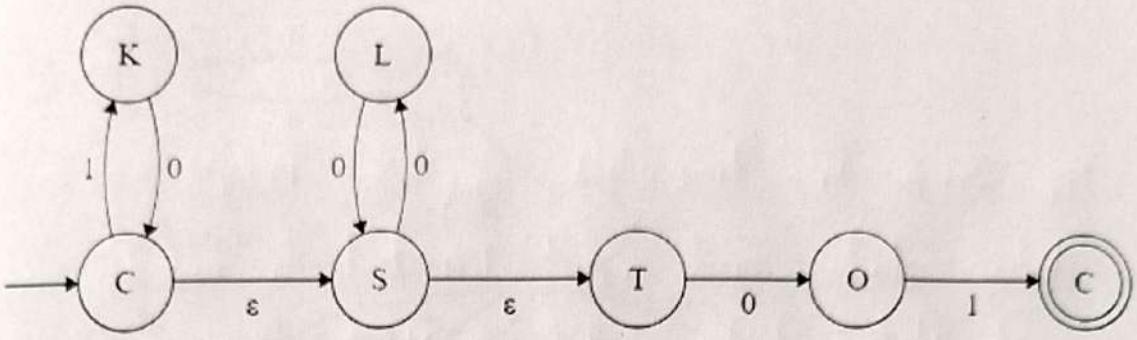


(or)



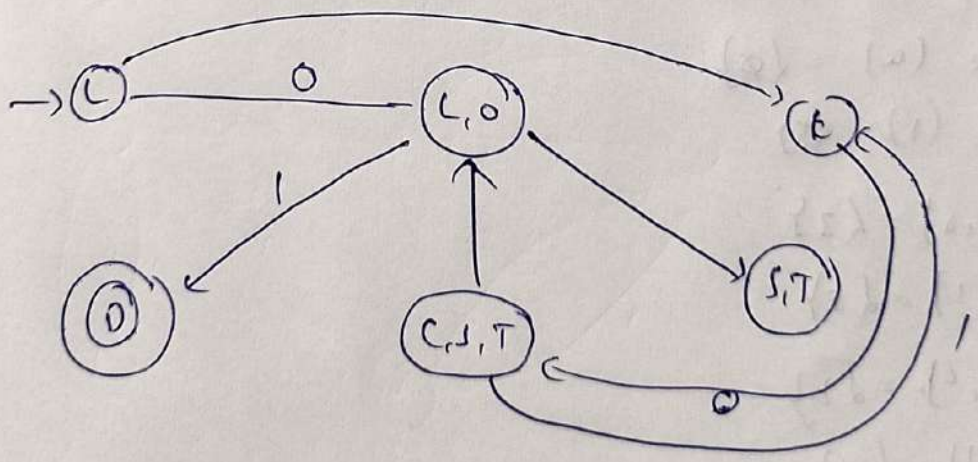
Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Convert the ε-NFA given in the form of a transition diagram to NFA without ε-transition.

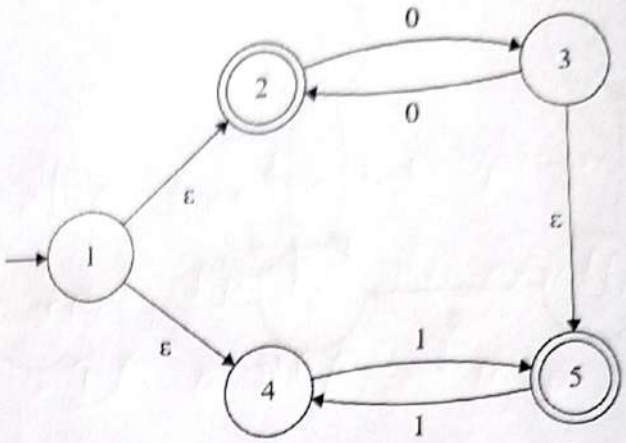


Solution:

	0	1
C	L, O	K
L, O	S, T	D
K	C S T	~~Ø~~
D	Ø	Ø
C S T	L, O	K



3. Transform the following ε-NFA to NFA.



Solution:

$Q = \{1,2,3,4,5\}$

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

$\epsilon\text{-close}(1) = \{1,2,4\}$

$\epsilon\text{-close}(2) = \{\emptyset\}$

$\epsilon\text{-close}(3) = \{3,5\}$

$\epsilon\text{-close}(4) = \{\emptyset\}$

$\epsilon\text{-close}(5) = \{5\}$

$\delta'(1,0) = \{3\}$

$\delta'(1,1) = \{5\}$

$\delta'(3,0) = \{2\}$

$\delta(3,1) = \{\emptyset\}$

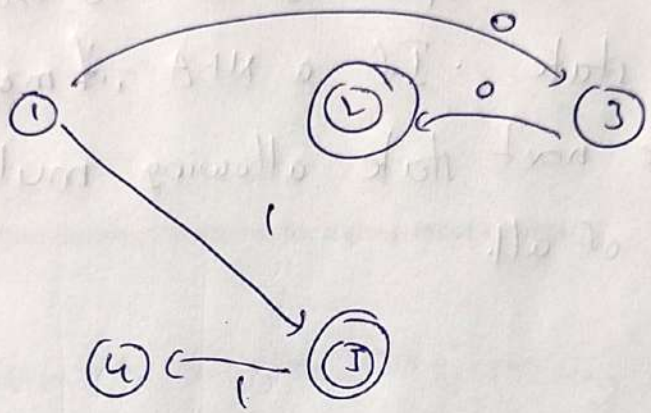
$\delta'(5,0) = \{\emptyset\}$ $\delta'(5,1) = \{4\}$

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

$\delta'(1,0) = \epsilon\text{closure}(1) = \{2,4\}$

$\delta'(1,1) = \epsilon\text{closure}(1) = \{5\}$

Diagram of NFA



Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva Questions

1. **How does an NFA differ from a DFA in terms of the transition function?**

Answer:

In DFA the transition function maps each ~~future~~ state and input symbol to exactly one next ~~state~~ state. If a NFA, it maps to set of possible next state allowing multiple transi or next of all.

2. **What is the purpose of ε-transitions in an NFA?**

Answer:

They allows the NFA to move b/w the without consuming any input symbolic provides more flexibility

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. How does an NFA determine whether an input string is accepted?

Answer:

An NFA accepts an input if there exists at least one path through the stated that leads to an accepting state.

4. Can an NFA have missing transitions for a given input symbol?

Answer:

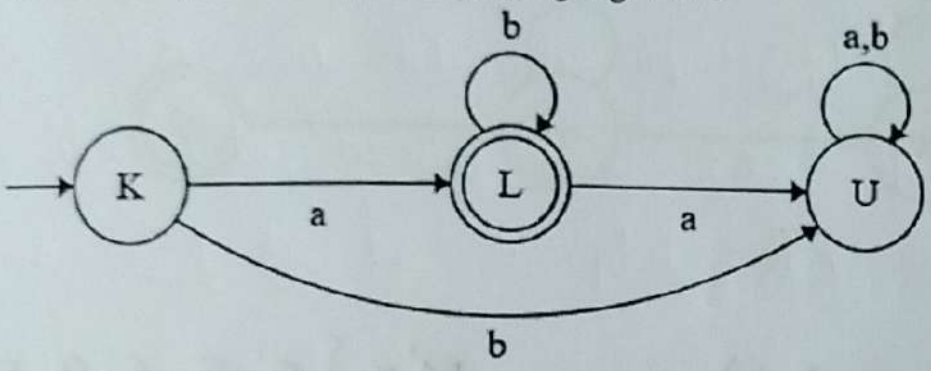
Yes, an NFA can have missing transition for some input symbols, It can still accepting the string if there of least one valid path to an accepting state.

Comment of the Evaluator (if Any)	**Evaluator's Observation** Marks Secured:_____out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Tutorial #	<TO BE FILLED BY STUDENT> 3	Student ID	2300030088
Date	20-8-24	Student Name	lejesh

Pre-Tutorial (To be completed by student before coming to tutorial class)

1. The following DFA recognizes the language ab^*.



Find an equivalent NFA for the above given DFA. Write each step.

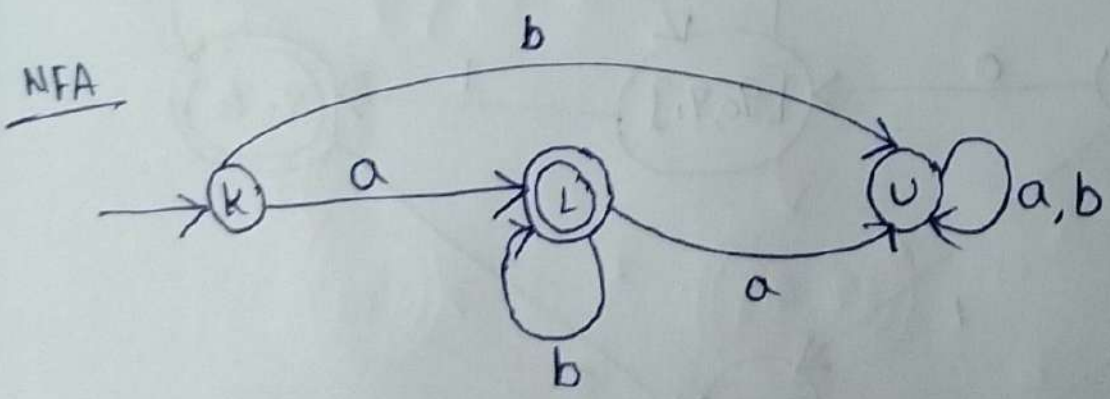
Solution: DFA

δ	a	b
→ K	L	U
L	U	L
* U	U	U

NFA

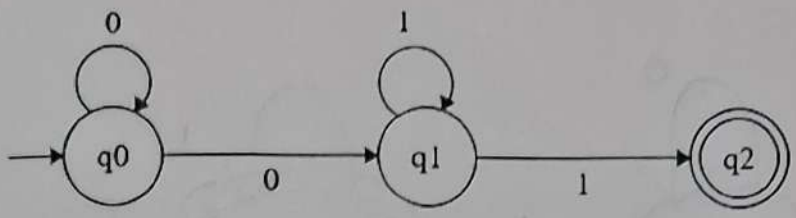
δ'	a	b
→ K	L	U
* L	U	L
U	U	U

NFA



Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Convert the following NFA to DFA.



Solution:

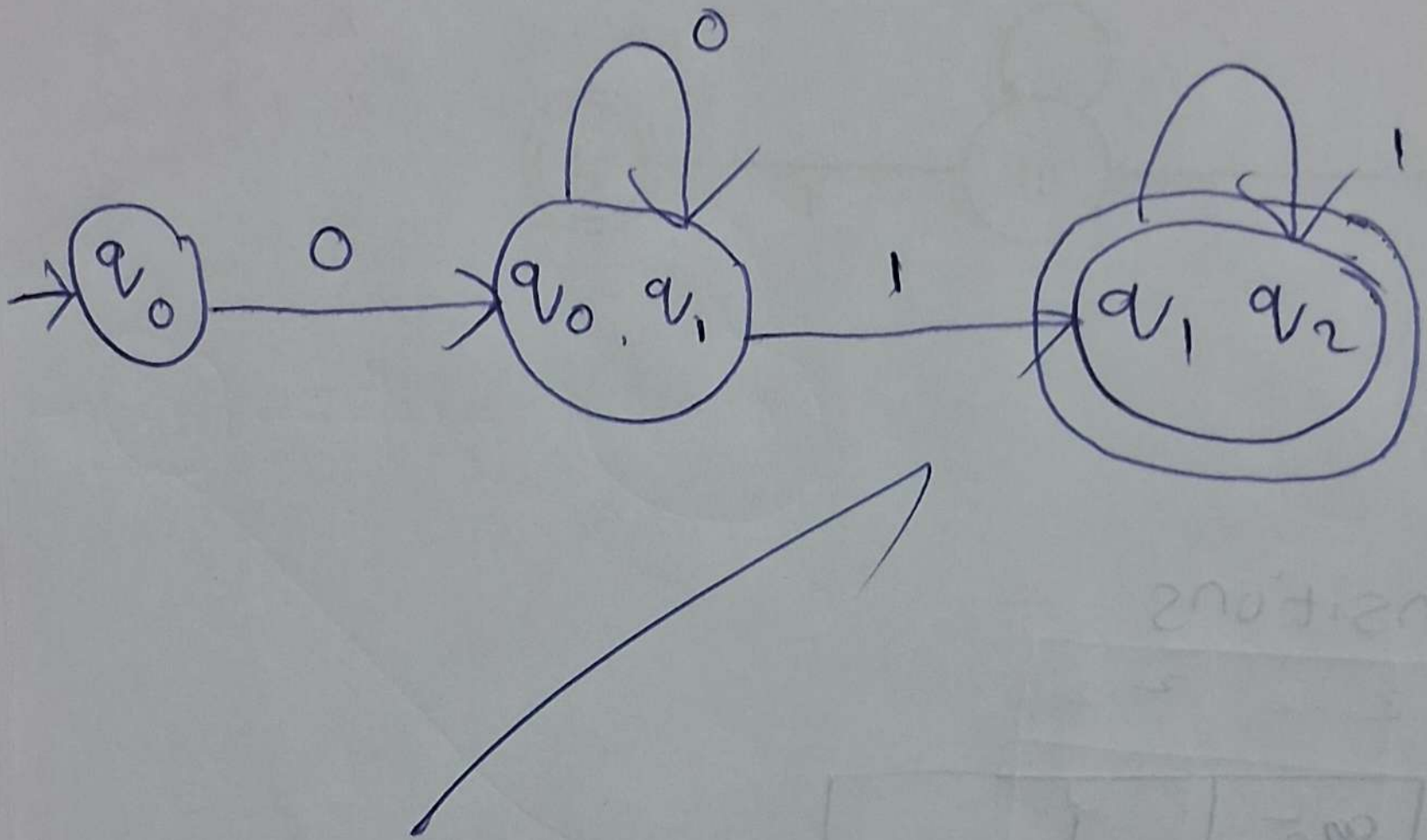
NFA Transitions

State	0	1
q_0	q_0, q_1	-
q_1	-	q_1, q_2
q_2	-	-

DFA Transitions

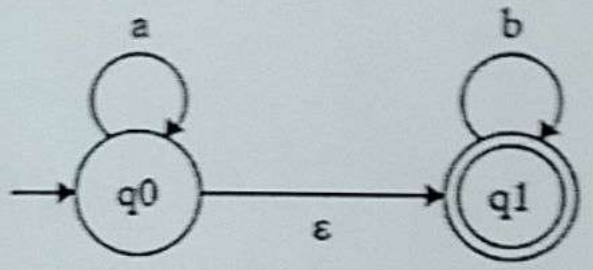
State	0	1
q_0	q_0, q_1	-
$q_0 q_1$	q_0, q_1	q_1, q_2
q_1, q_2	-	q_1, q_2

DFA diagram



Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Find the language recognized by the following NFA.



Construct an equivalent DFA. Write each step.

Solution:

$L = \{ab, aab, aabb, aabbb, \ldots\}$

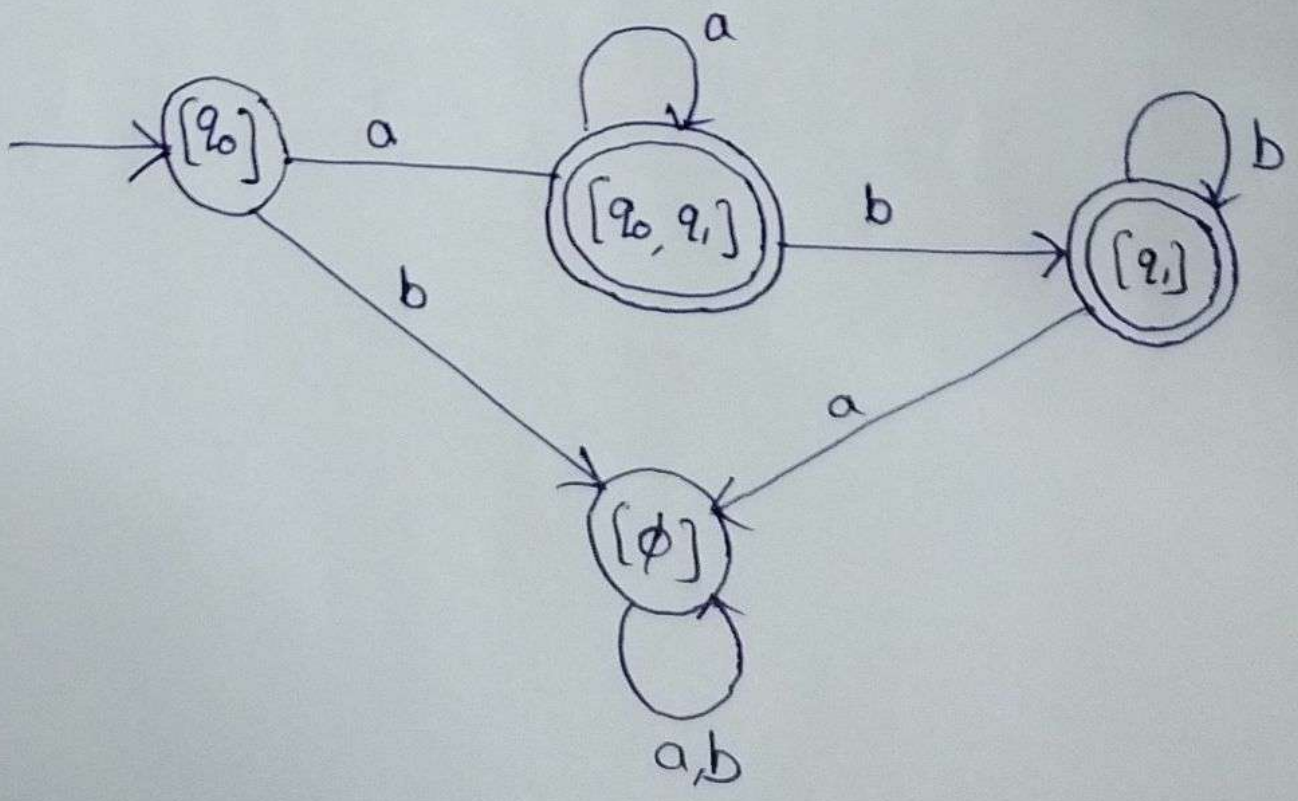
$L = \{\omega / \omega \text{ Contains a's followed by any No. of b's}\}$

NFA

δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	ϕ
$*q_1$	ϕ	$\{q_1\}$

DFA

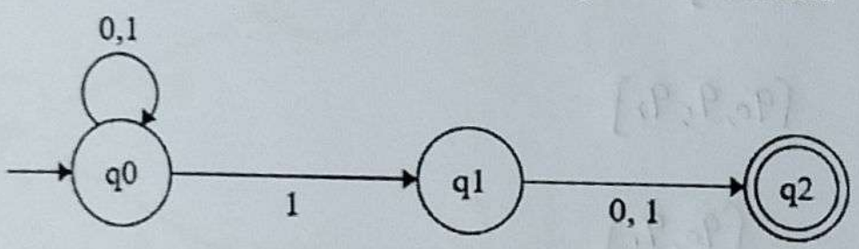
δ'	a	b
$\rightarrow [q_0]$	$[q_0, q_1]$	$[\phi]$
$*[q_0, q_1]$	$[q_0, q_1]$	$[q_1]$
$*[q_1]$	ϕ	$[q_1]$



Tutorial #	3	Student ID	2300030088
Date	20-8-24	Student Name	B. Tejesh

IN-TUTORIAL (To be carried out by student in presence of faculty in classroom)

1. Find the language recognized by the following finite automaton.



Construct an equivalent DFA for the above given automaton.

Solution:

$L = \{011, 111, 010, 11100, ---\}$

$L = \{\omega | \omega$ Contains any no. of zeroes or one's followed by 1 any no. of zeroes or one's$\}$

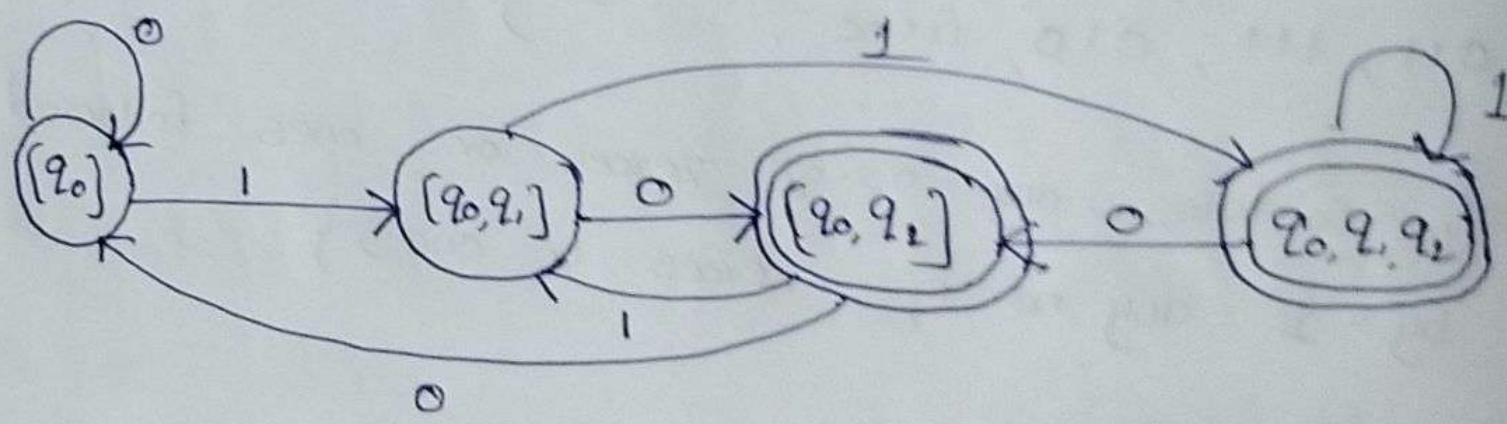
NFA

δ	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
$* q_2$	ϕ	ϕ

Tutorial #	<TO BE FILLED BY STUDENT> 3	Student ID	<TO BE FILLED BY STUDENT> 2
Date	<TO BE FILLED BY STUDENT> 20-8-24	Student Name	<TO BE FILLED BY STUDENT> B. Rajesh

DFA

S'	0	1
$[q_0]$	$[q_0]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1, q_2]$
* $[q_0, q_2]$	$[q_0]$	$[q_0, q_1]$
* $[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_0, q_1, q_2]$

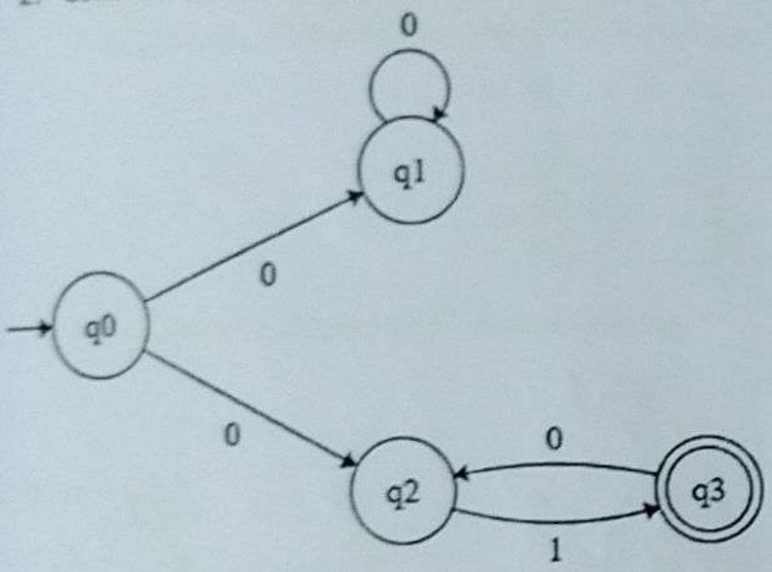


Course Title	THEORY OF COMPUTATION
Course Code(s)	23MT2014

ACADEMIC YEAR:

Tutorial #	3	Student ID	2300030088
Date	20-8-24	Student Name	B. Tejesh

2. Consider the following NFA that recognizes language L.



Construct an equivalent DFA for the above NFA. Write each step.

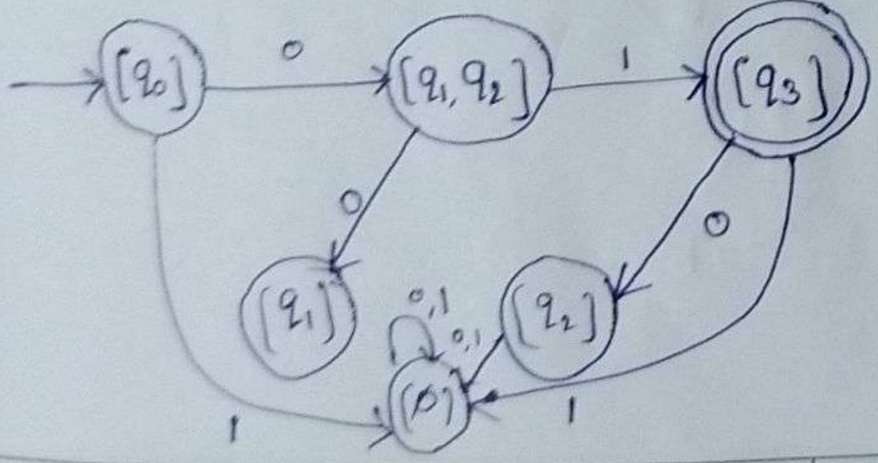
Solution:

NFA

δ	0	1
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$
q_1	$\{q_1\}$	$\emptyset$
q_2	$\emptyset$	$\{q_3\}$
$*q_3$	$\{q_2\}$	$\emptyset$

DFA

δ	0	1
$\rightarrow [q_0]$	$[q_1, q_2]$	$\emptyset$
$[q_1, q_2]$	$[q_1]$	$[q_3]$
$* [q_3]$	$[q_2]$	$\emptyset$
$[q_1]$	$[q_1]$	$\emptyset$
$[q_2]$	$\emptyset$	$[q_3]$



Tutorial #	3	Student ID	2300030088
Date	20-8-24	Student Name	B Yogesh

3. Consider an NFA with three states, $Q = \{q0, q1, q2\}$, two input symbols, $\Sigma = \{a, b\}$, and $F = \{q1, q2\}$. The transition function is defined as follows:

	a	**b**
q0	q1	q2
q1	∅	q1
q2	q2	∅

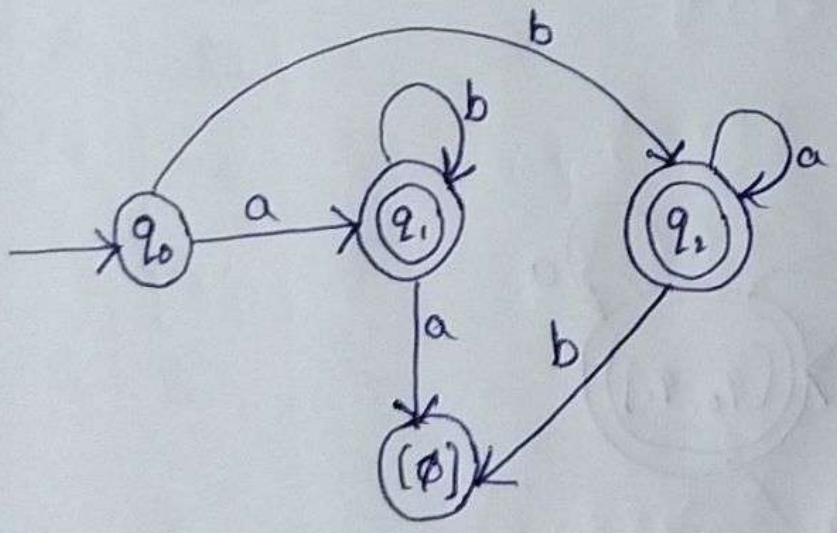
Construct an equivalent DFA? Write each step.

Solution: NFA

δ	a	b
→ q_0	$\{q_1\}$	$\{q_2\}$
* q_1	∅	$\{q_1\}$
* q_2	$\{q_2\}$	∅

DFA

δ'	a	b
→ q_0	$[q_1]$	$[q_2]$
* q_1	$[\phi]$	$[q_1]$
* q_2	$[q_2]$	$[\phi]$



Post-Tutorial (To be carried out by student after Tutorial session is over)

1. Consider an NFA with two states, $Q = \{q0, q1\}$, two input symbols, $\Sigma = \{a, b\}$, and F =
   The transition function is defined as follows:

	a	b
$q0$	$q0$	$q1$
$q1$	$\{q0, q1\}$	$\emptyset$

Construct an equivalent DFA. Write each step.

Solution:

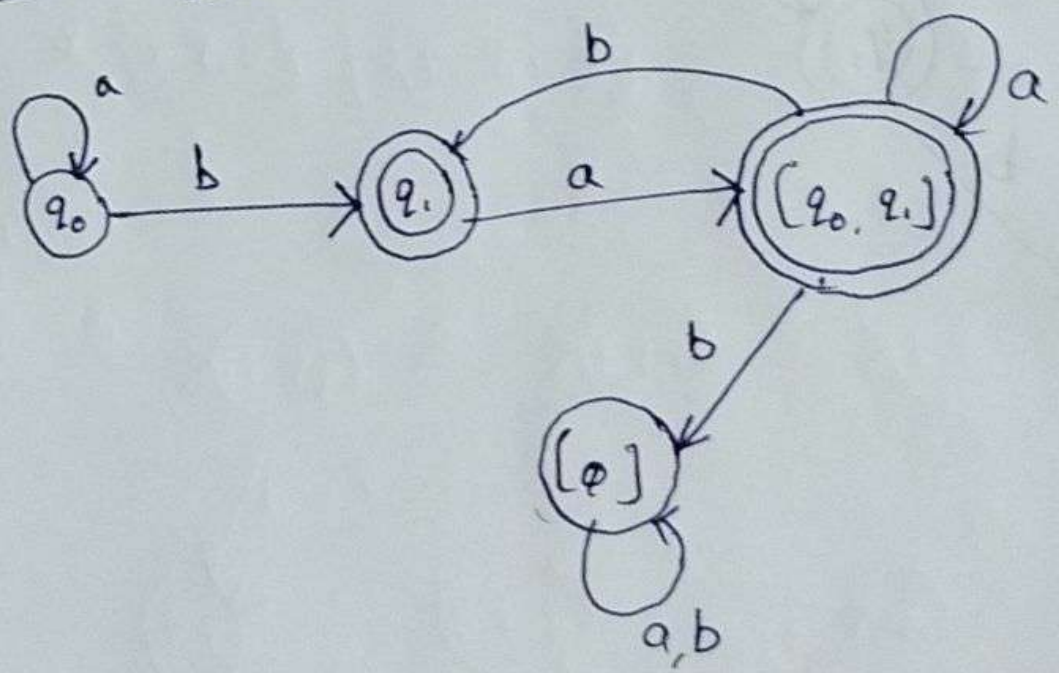
NFA

δ	a	b
$\rightarrow q_0$	$\{q_0\}$	$\{q_1\}$
$* q_1$	$\{q_0, q_1\}$	$\emptyset$

DFA

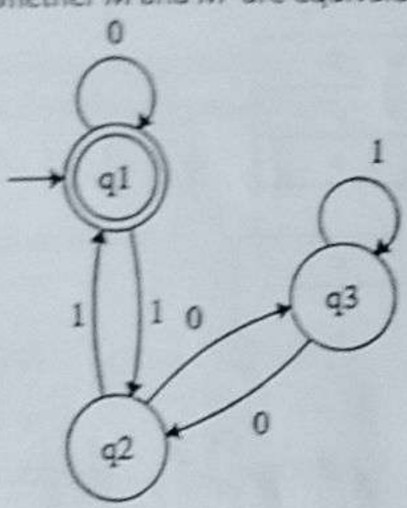
δ	a	b
$\rightarrow q_0$	$[q_1]$	$[q_1]$
$* q_1$	$[\phi]$	$[q_1]$
$*[q_0 q_1]$	$[q_1 \phi]$	$[q_1]$

Result DFA

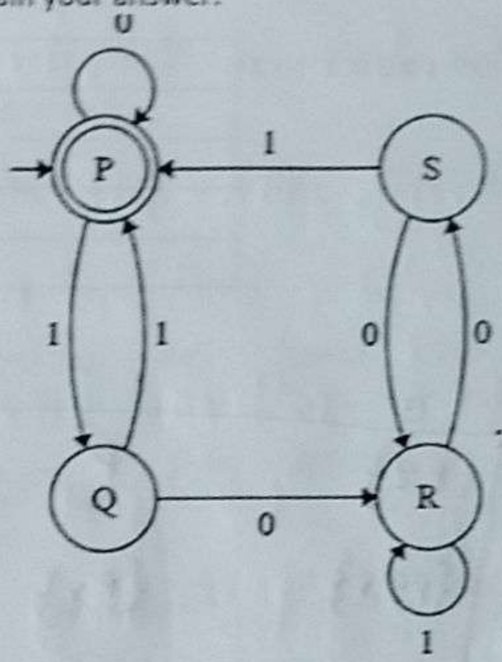


Tutorial #	3	Student ID	2300030088
Date	20-8-24	Student Name	G. Tejesh

2. Consider the following two DFAs M and M' over {0, 1} given in in following figure. Determine whether M and M' are equivalent. Explain your answer.



M



M'

Solution:

DFA : M

δ	0	1
$*\to q_1$	q_1	q_2
q_2	q_3	q_1
q_3	q_2	q_3

DFA M'

δ'	0	1
$*\to P$	P	Q
Q	R	$\emptyset$
R	S	R
S	R	P

Let map states from M to M':

$q_1 \to P$

$q_2 \to Q$

$q_3 \to S$

$q_1 \to 0 \to q_1$ maps to $P \to 0 \to P$

$q_1 \to 1 \to q_2$ maps to $P \to 1 \to Q$

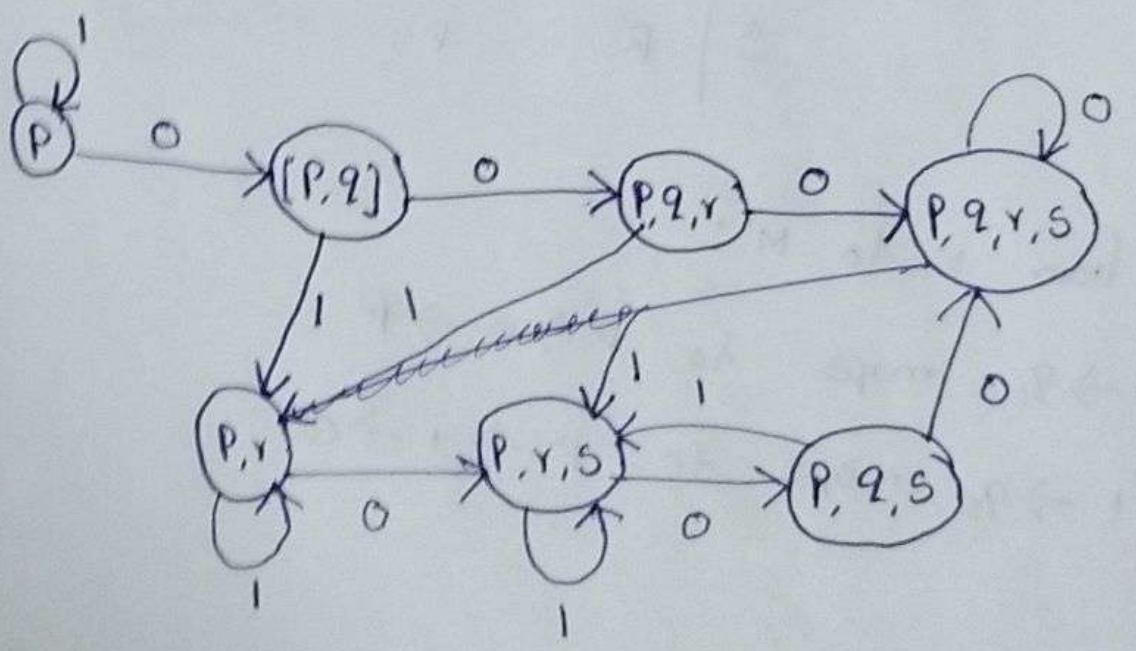
Tutorial #	3	Student ID	2800039008
Date	20-8-24	Student Name	Jegesh

3. Convert the following NFA to its equivalent DFA where ({p, q, r, s}, {0,1}, d, p, {s}} where d is

d	0	1
p	p, q	p
q	r	r
r	s	-
s	s	s

Solution:

	0	1
→ p	{p,q}	p
{p,q}	{p,q,r}	{p,r}
{p,q,r}	{p,q,r,s}	{p,r}
*{p,q,r,s}	{p,q,r,s}	{p,r,s}
{p,r}	{p,q,s}	{p,r}
*{p,r,s}	{p,q,s}	{p,r,s}
*{p,q,s}	{p,q,r,s}	{p,r,s}



Tutorial #	3	Student ID	2300030088
Date	20-8-24	Student Name	Tejesh

Viva Questions:

4. What is the equivalence theorem for NFAs and DFAs? How does it ensure that any NFA can be converted to an equivalent DFA, and vice versa?

Answer:

Equivalence Theorem :-

NFA and DFA'S recognize the same set of languages (regular languages). Any NFA can be converted to an equivalent DFA by Simulating all possible Computation paths Simultoneously using DFA States repesenting set of NFA states.

Tutorial #	<TO BE FILLED BY STUDENT> 3	Student ID	23000300088
Date	<TO BE FILLED BY STUDENT> 20-8-24	Student Name	<TO BE FILLED BY STUDENT> Jagesh

5. How is the equivalence of NFAs and DFAs used in compiler design? Provide an example of how a compiler uses this equivalence to recognize and parse programming languages.

Answer: Equivalence in Compiler design :-

Compiler leverage NFA - DFA equivalence to Construct efficient lexical analyzers. NFA's are used to model token patterens flexibly. then Converted to DFA's for fast and deterministic token recognition during lexical analysis.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Tutorial #	4	Student ID	23000555
Date	20-8-24	Student Name	[illegible]

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Let $r = a(a + b)^*$, $s = aa^*b$ and $t = a^*b$ be three regular expressions. Provide the relationship among the languages $L(r)$, $L(s)$, and $L(t)$. Explain.

Solution:

$r = a(a+b)^*$

$s = aa^*b$

$t = a^*b$

L(r): language of r Consists of all strings that start with an a followed by any Combination of as and bs

L(s) : consists of all strings that start with one (or) more No. of a's followed by a single b

L(t): consists of all the strings start with zero (or) any No. of a's followed by a single b.

Relationship:—

1) L(s) is a subset of L(r) and L(t)

2) L(t) is not a subset of L(r)

∴ L(s) is a Common set of both L(r) and L(t)

Tutorial #	4	Student ID	230000300088
Date	20-8-24	Student Name	Tejesh

2. Write the regular expression corresponding to the language L where L = { x ∈ {0, 1}* | x ends with 1 and does not contain substring 00 }. Explain.

Solution:

→ End with 1

→ Does not Contain 00

$L = (1+01)^*1$

$L = \{1, 11, 101, 1101, \ldots\}$

3. Prove that $a(ba)^*b = ab(ab)^*$

Solution:

The string accepted by or Can be written as $a(ba^*)b$

is $ab + abab + ababab + \ldots$

$ab(ab)^* = ab + abab + ababab + \ldots$

The Expanded form the strings accepted are equal

$\therefore a(ba)^* b = ab(ab)^*$

Tutorial #	<4 BE FILLED BY STUDENT>	Student ID	<2300030088
Date	<20-8-24 BY STUDENT>	Student Name	<B. Jagesh

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Prove that $(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1) = 0^*1(0 + 10^*1)^*$

Solution:

LHS = $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

$=(1+00^*1)[\epsilon+(0+10^*1)^*(0+10^*1)]$

$=(1+00^*1)(0+10^*1)^*$

$=1[\epsilon+00^*][0+10^*1)^*$

$=10^*(0+10^*1)^*$

$=0^*1(0+10^*1)^*$

Hence proved LHS = RHS

Tutorial #	<TO BE FILLED BY STUDENT> 4	Student ID	2300030088
Date	<TO BE FILLED BY STUDENT> 20-8-24	Student Name	B. Tejesh

2. Give **TWO** R.E.s for representing the set L of strings in which every 0 is immediately followed by at least two 1s.

Solution: (i) $R_1 = (1^*)(011)^*(1^*)$

(ii) $R_2 = (1^*(011^*)1^*)$

3. Prove $(a+b)^* = a^*(ba^*)^*$ using identity rules of regular set.

Solution:

$$(a+b)^* = \epsilon + (a+b) + (a+b)(a+b) + \cdots$$

$$= \epsilon + a + b + aa + ab + ba + bb + \cdots$$

$$= (\epsilon + a + aa + \cdots) + (b + ab + \cdots)$$

$$= a^* + (b + ab + \cdots)$$

$$= a^* + b(\epsilon + a + \cdots)$$

$$= a^* + ba^*$$

$$= a^*(\epsilon + ba^*)$$

$$= a^*(ba^*)$$

LHS = RHS

Hence Proved.

Tutorial #	4	Student ID	23000300[illegible]
Date	20-8-24	Student Name	Yegesh

Post-Tutorial (To be carried out by student after attending the tutorial session)

1. Find the regular expression representing the set of all strings of the form $a^m b^n c^p$ *where* $m, n, p \geq 1$

Solution:

$L = \{abc, aabbcc, aaabbbccc, ----\}$

$= (a^* a)(b^* b)(c^* c)$

2. Find the sets represented by the regular Expression $(aa)^* + (aaa)^*$

Solution:

The sets are $\{aa, aaaa, ----, aaa, aaaa, ---\}$

3. Prove that $P + PQ^*Q = a^*bQ^*$ where $P = b + aa^*b$ and Q is any regular expression.

Solution:

$\Rightarrow (b + aa^*b) + (b + aa^*b)Q^*Q = a^*b^*Q^*$

$\Rightarrow (b + aa^*b) + (b + QQ^*b)Q^*Q = bQ^* + aa^*bQ^*$

$\Rightarrow a^*bQ^*$

Tutorial #	<TO BE FILLED BY STUDENT> 4	Student ID	<TO BE FILLED BY STUDENT> 2300030088
Date	<TO BE FILLED BY STUDENT> 20-8-24	Student Name	<TO BE FILLED BY STUDENT> B. Legesh

Viva Questions

1. What is the relationship between regular sets and regular expressions? How do they help us describe and analyze patterns in strings?

Answer: Regular sets are the languages described by regular expression and they represent patterns in string that can be matched

2. How do we determine whether two regular expressions are equivalent, and what are the implications of equivalence in terms of language recognition and automata theory?

Two regular expression are equivalent if they describe the same regular language which means the generate (or) match the same set of strings.

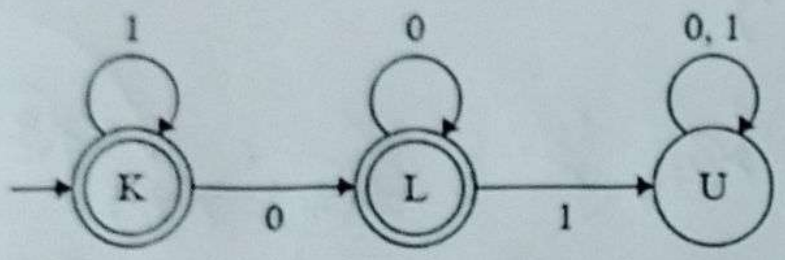
(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Tutorial #	5	Student ID	2300032088
Date	20/8/24	Student Name	Yogesh

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Construct the regular set accepted by the finite automaton whose transition diagram is as shown below.



Solution:

Equation:

$K = E + K_1 \rightarrow (1)$

$L = K0 + L0 \rightarrow (2)$

$U = L1 + U0 + U1 \rightarrow (3)$

from (1) $\Rightarrow \underset{R}{K} = \underset{Q}{E} + \underset{RP}{K1}$

$K = E1^* = 1^*$

$L = 1^*0 + L0$

$R = Q \quad RP$

$L = 1^*00^*$

R.E for given DFA is $= 1^* + 1^*00^*$

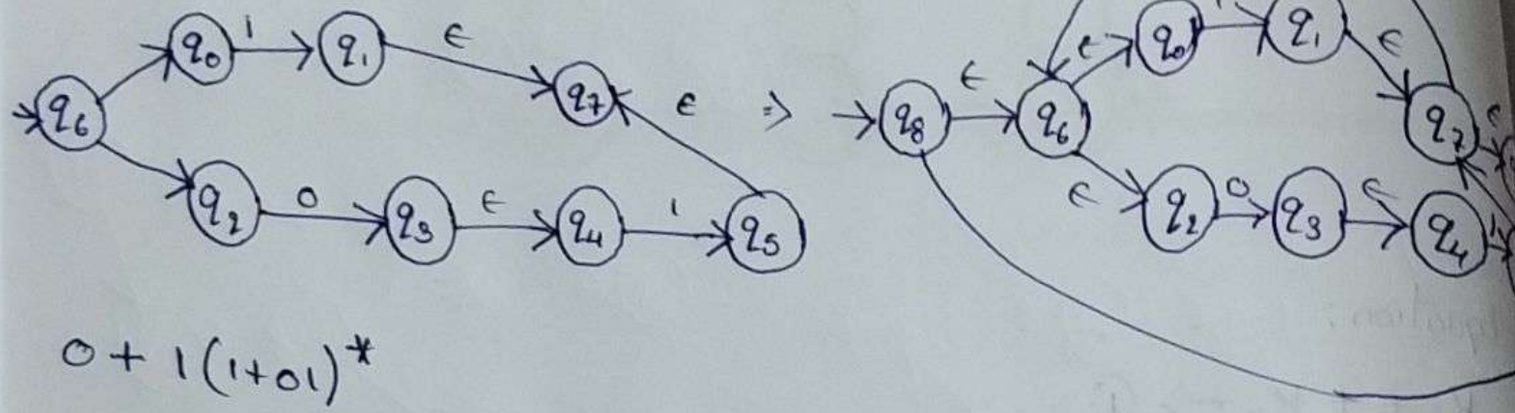
Regular set $= \{1, 11, 1111, 1110000, \ldots\}$

Tutorial #	5	Student ID	23088
Date	2018	Student Name	Tejesh

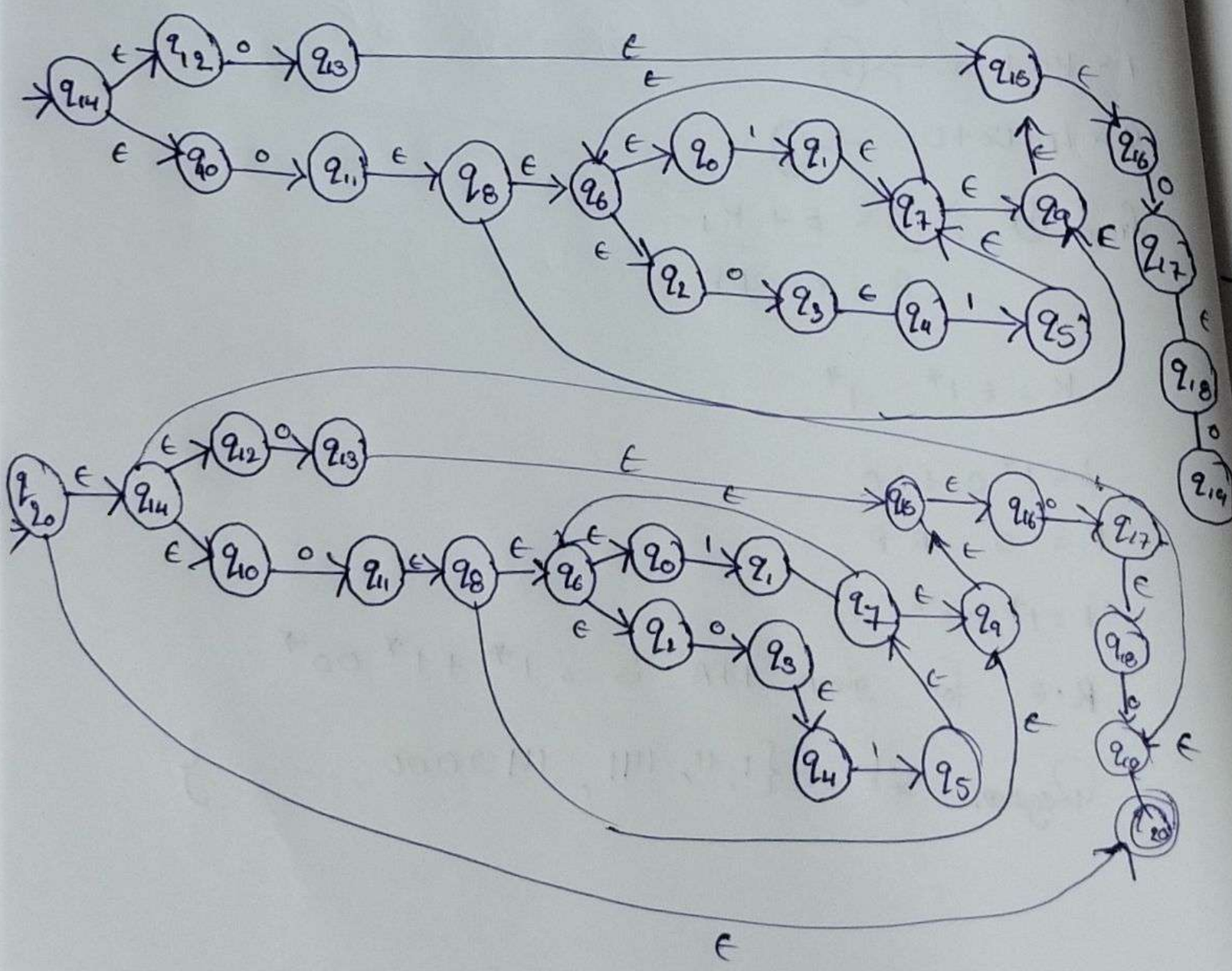
2. Construct a NFA for the regular expression (0 + 1(1 + 01)* 00)* Write each step.

Solution: $(0 + 1(1+01)^* 00)^*$

$(1+01)^* \Rightarrow (1+01)$



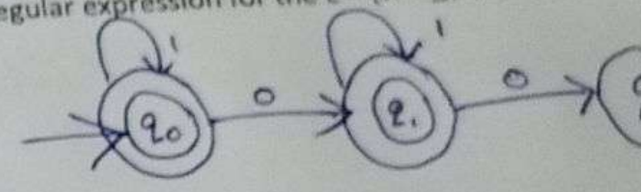
$0 + 1(1+01)^*$



Tutorial #	5	Stude
Date	2018	Student

3. Write the regular expression for the L = {w ∈ (0,1)* | w has

Solution:



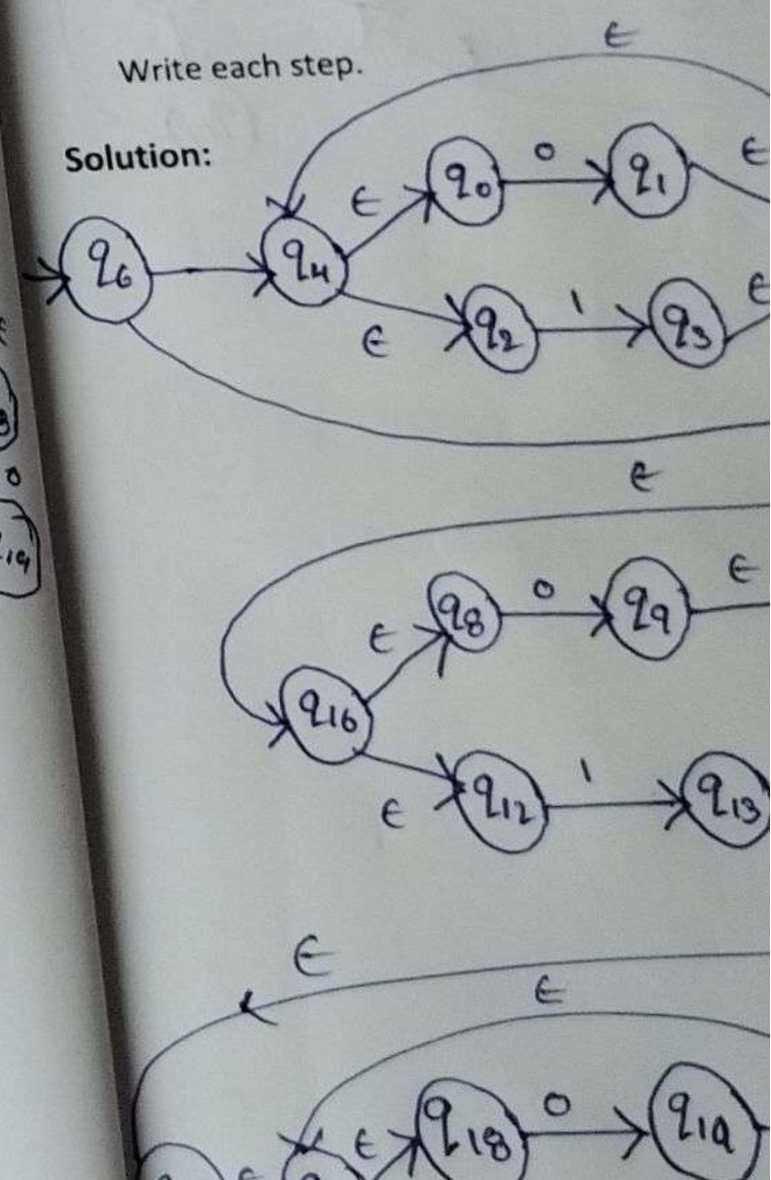
$(1+01)^*$

IN-TUTORIAL (To be carried out in presen

1. Construct the finite automaton equivalent to
$(0 + 1)^*(00 + 11)(0 + 1$

Write each step.

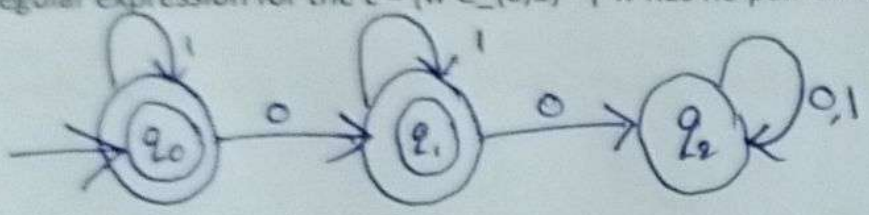
Solution:



Tutorial #	5	Student ID	300988
Date	2018	Student Name	Tejesh

3. Write the regular expression for the L = {w ∈_{0,1}* | w has no pair of consecutive zeros?

Solution:



$(1+01)^*$

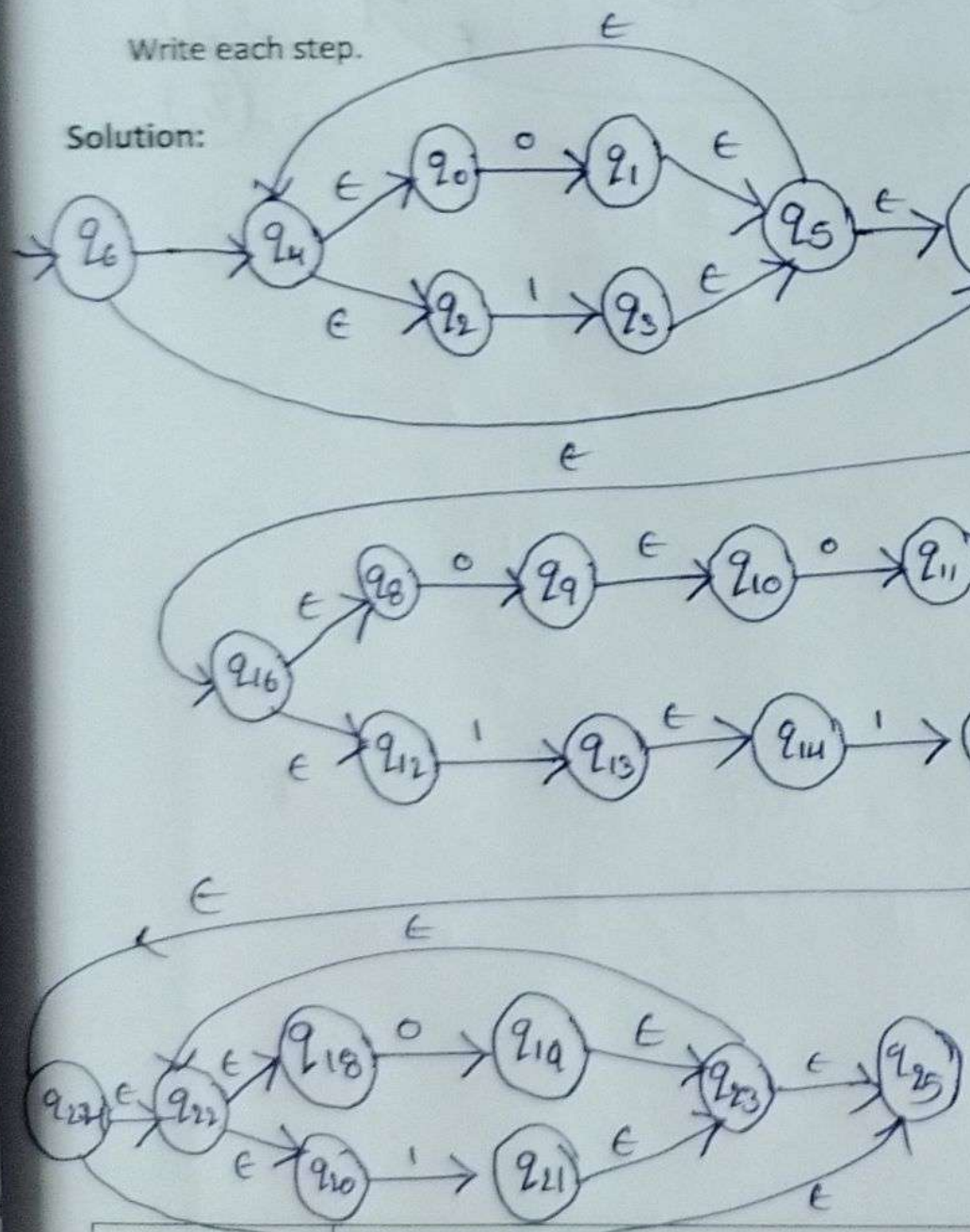
IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Construct the finite automaton equivalent to the regular expression

$$(0 + 1)^*(00 + 11)(0 + 1)^*$$

Write each step.

Solution:

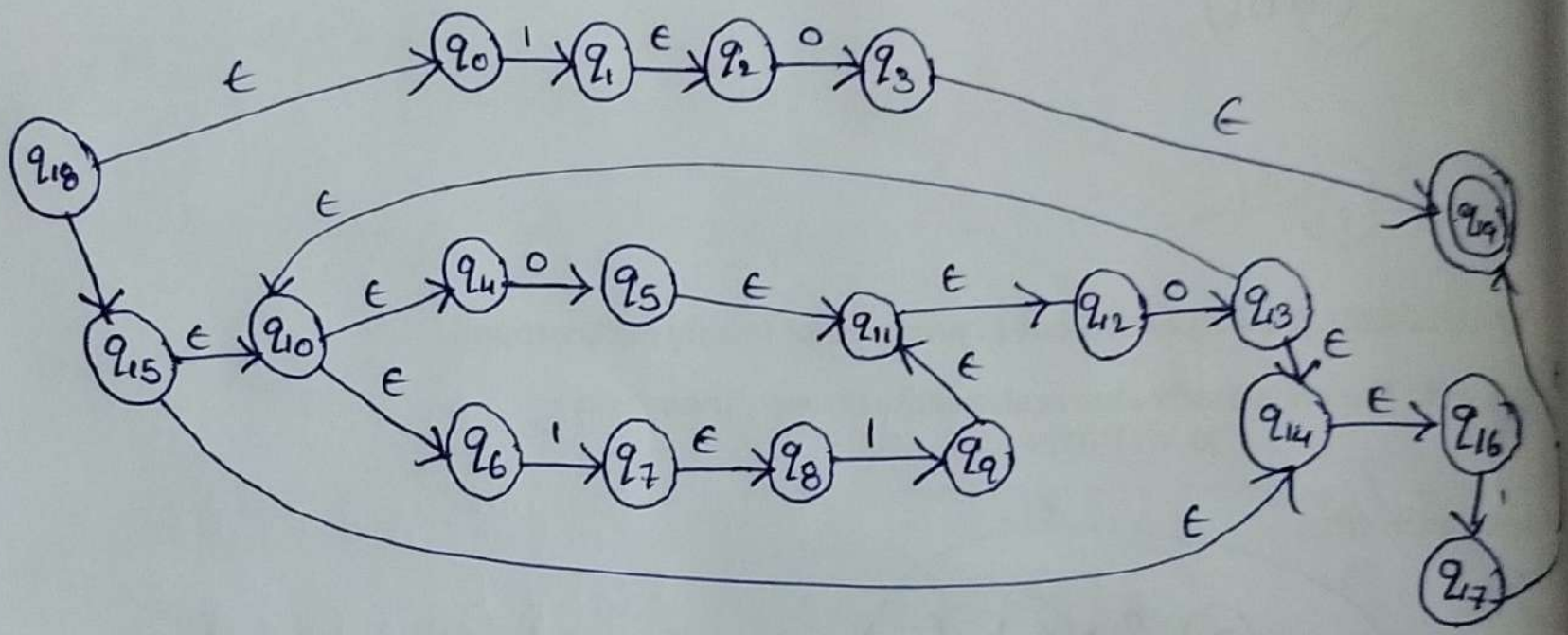


Tutorial #	<TO BE FILLED BY STUDENT> 5	Student ID	<TO BE FILLED BY STUDENT> 30088
Date	<TO BE FILLED BY STUDENT> 20/8	Student Name	<TO BE FILLED BY STUDENT> Tejesh

2. Construct a DFA with reduced states equivalent to the regular expression $10 + (0 + 11)0^*$

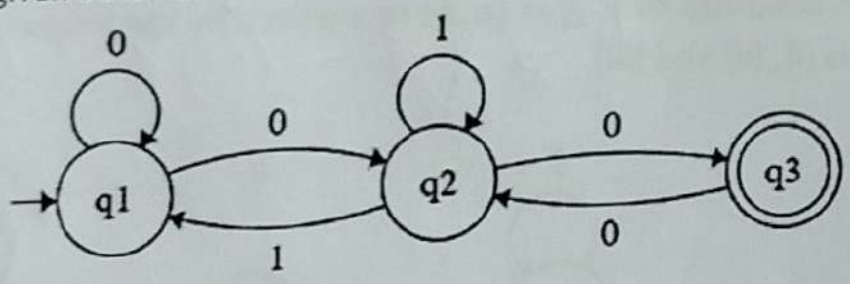
3.

Solution:



Tutorial #	5	Student ID	30088
Date	20/8	Student Name	Tejesh

3. Consider the following transition diagram of a finite automaton. Prove that the strings recognized by this given automaton is **(0 + 0(1 + 00)*1)* 0(1 + 00)* 0.**



4.

Solution: $(0+0(1+00)^*1)$: Represents the first part of the string, which consists of zero or more repetitions of either a single 0(1) a sequence of 0's followed by a

→ This corresponds to the transition blw q_1 and q_2

→ 0: Represents mandatory zero

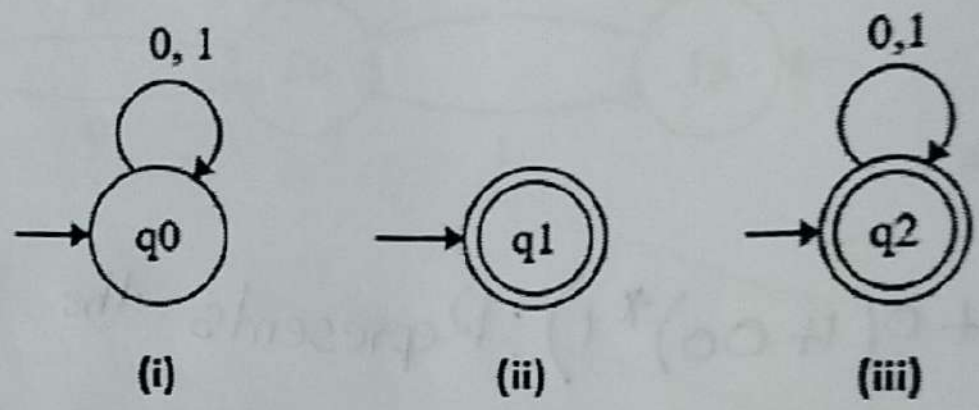
→ $(1+00)^*$: Represents 1 (or) 3 or 0 followed by zero

→ 0; Represents the final 0 in string.

Tutorial #	<TO BE FILLED BY STUDENT> 5	Student ID	<TO BE FILLED BY STUDENT> 30088
Date	<TO BE FILLED BY STUDENT> 2018	Student Name	<TO BE FILLED BY STUDENT> Tejesh

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Find the set of strings over $\sum = \{a, b\}$ recognized by the following transition diagrams/automatons (i), (ii) and (iii).



Solution:

(i) No strings are accepted because there is no Final state

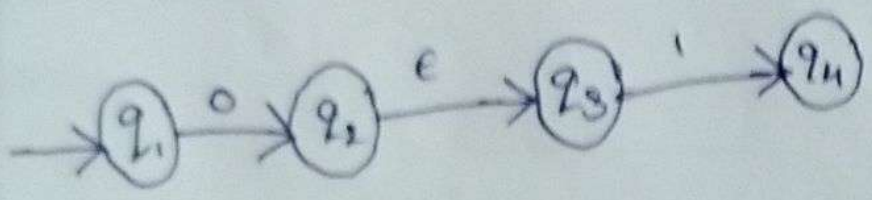
(ii) { } (or) ϵ

(iii) {0, 1, 001, 000, 011, ---}

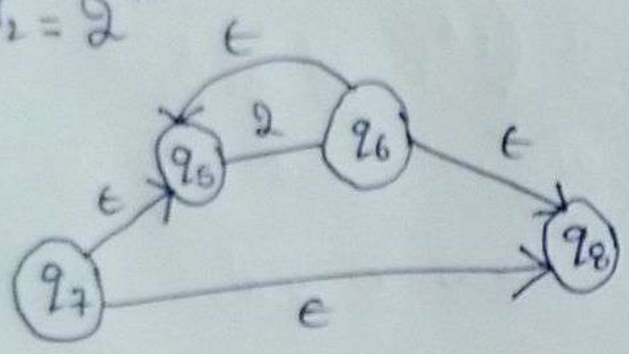
Tutorial #	5	Student ID	230388
Date	20/8	Student Name	Tejesh

2. Construct a finite automaton for the regular expression (01 + 2*)*1

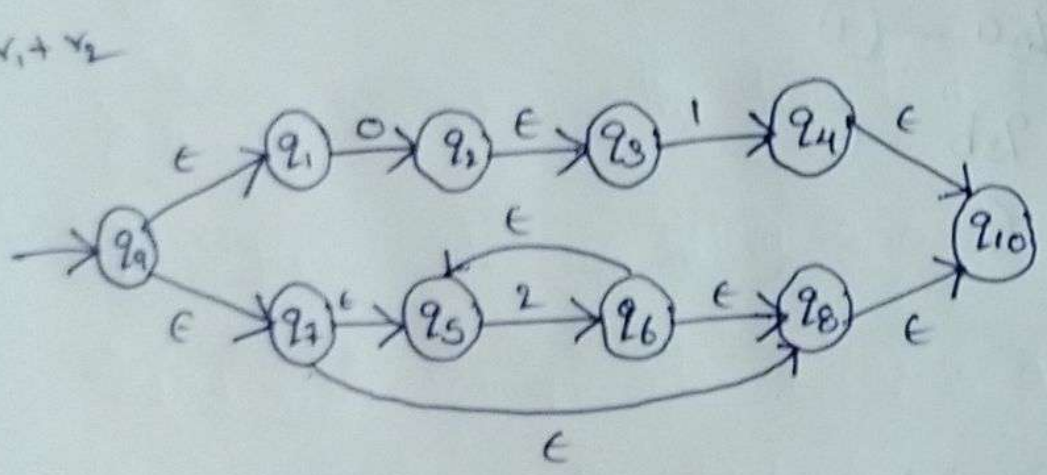
Solution: $Y_1 = 01$



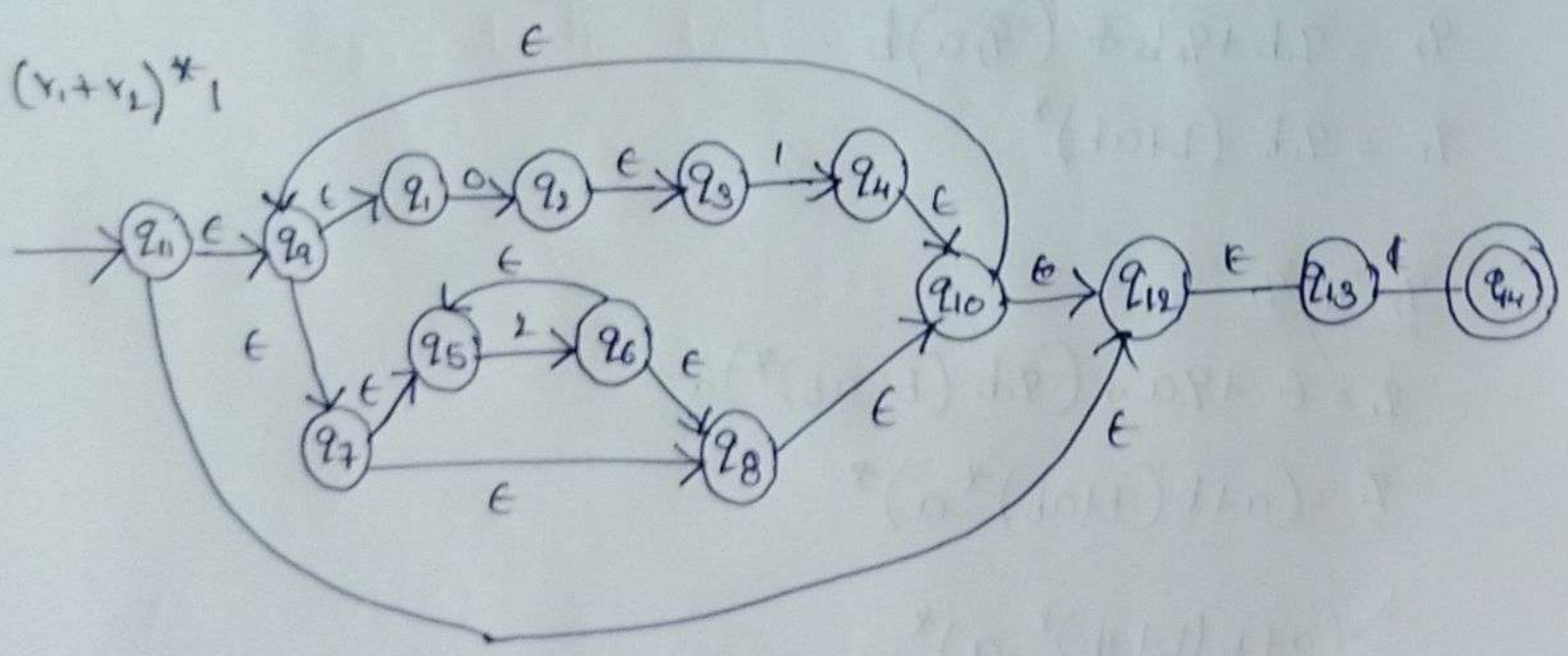
$Y_2 = 2^*$



$Y_1 + Y_2$

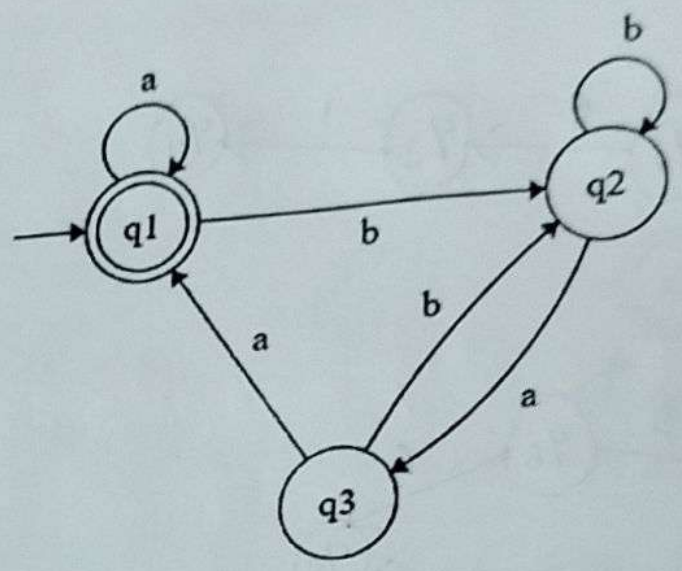


$(Y_1 + Y_2)^* 1$



Tutorial #	<TO BE FILLED BY STUDENT> 5	Student ID	<TO BE FILLED BY STUDENT> 300888
Date	<TO BE FILLED BY STUDENT> 2018	Student Name	<TO BE FILLED BY STUDENT> Tejesh

3. Construct a regular expression corresponding to the state diagram described by the following figure.



Solution:

$q_1 = \epsilon + q_1 a + q_3 a$ — (1)

$q_2 = q_1 b + q_2 b + q_3 b$ — (2)

$q_3 = q_2 a$ — (3)

q_3 in q_2

$q_2 = q_1 b + q_2 b + (q_2 a) b$

$q_2 = q_1 b (b + ab)^*$

q_2 in q_1

$q_1 = \epsilon + q_1 a + (q_1 b (b + ab)^*) a$

$q_1 = (a + b(b + ab)^* a)^*$

$\Rightarrow (a + b(b + ab)^* a)^*$

Tutorial #	5	Student ID	30088
Date	2018	Student Name	Tejesh

Viva Questions:

1.How does the structure of the NFA produced by Thompson's construction technique differ from the structure of the original regular expression?

Solution: i) Graphical Vs Linear

The NFA is a graphical structure with states and transitions, whereas there is a linear, and transitions coh Symbolic representation

ii) Epsilon Transitions:—

The NFA Explicitly represents transition and intermediate states, while the R.E implies them through operation

iii) Size

The NFA is generally larger than R.E

Tutorial #	<TO BE FILLED BY STUDENT> 5	Student ID	2300030088
Date	20-8-24	Student Name	Tejesh

2.How does the structure of the regular expression produced by the state elimination method differ the structure of the original DFA?

Solution:

i) DFA : Represents the language as a sequential process of state transitions based on input symbols.

ii) R.E : Represents the language as a Combination of Symbols and operators.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____ out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Tutorial #	<TO BE FILLED BY STUDENT> 6	Student ID	<TO BE FILLED BY STUDENT> 23005
Date	<TO BE FILLED BY STUDENT> 16/8/24	Student Name	<TO BE FILLED BY STUDENT> G.VG.Mani

TUTORIAL SESSION 06:

DFA Minimization

Concept Building

Construct an automaton with the minimum number of states equivalent to a given a
With fewer states and transitions, the minimized DFA requires fewer computations and mem
resulting in faster execution.

Definition Two states q_1 and q_2 are equivalent (denoted by $q_1 \equiv q_2$) if both $\delta(q_1,x)$
final states. or both are nonfinal states for all $x \in \sum^*$ .As it is difficult to construct $\delta(q_1,$
for all $x \in \sum^*$ (there are an infinite number of strings in $\sum^*$). we give one more definition.

Definition Two states q_1 and q_2 are k-equivalent $(k \geq 0)$ if both $\delta(q_1,x)$ and $\delta(q_2,x)$
or both nonfinal states for all strings $|x| \leq k$.

Myhill-Nerode Theorem

It states that a regular language can be recognized by a deterministic finite automaton (DFA)
be recognized by a non-deterministic finite automaton (NFA) with a finite number of states.
provides a method for minimizing a DFA by eliminating unnecessary states and transitions, re
efficient and simplified automaton.

Steps for Minimization:

Step 1: Create a Table: Create a table for all pairs of states in the DFA.

Step 2: Mark Pairs: Mark all pairs of states where one state is final and the other is n

Step 3: Repeat: Repeat Step 2 until no more pairs can be marked.

Step 4: Combine: Combine all unmarked pairs of states into a single state in the minir

Tutorial #	<TO BE FILLED BY STUDENT> 6	Student ID	<TO BE FILLED BY STUDENT> 3148
Date	<TO BE FILLED BY STUDENT> 1/8/24	Student Name	<TO BE FILLED BY STUDENT> G.V.G. Haneesh

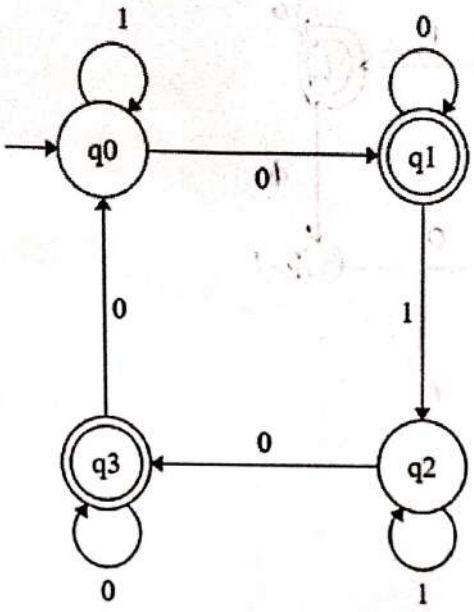
Pre-Tutorial (To be completed by student before attending tutorial session)

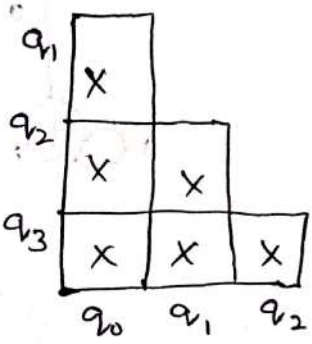
1. Find the equivalent states in the following automaton. Explain.

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

$F = \{q_1, q_3\}$

$Q - F = \{q_0, q_2\}$





After Iteration 1
$(q_2, q_0) \Rightarrow$ empty

After Iteration
No empty.

Solution:

(i) Equivalent States:

Reason: There are no equivalent states in this DFA, Because the given DFA is treated as minimal DFA

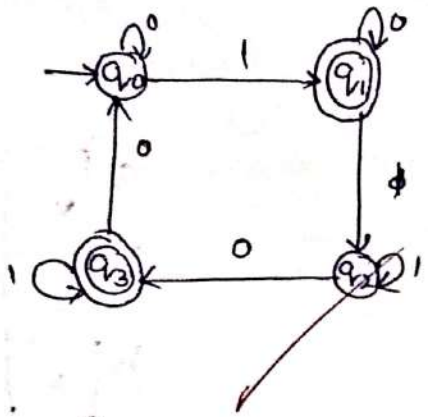
(ii) Non-equivalent states:

Reason: All states $Q = \{q_0, q_1, q_2, q_3\}$ are non-equivalent states

Tutorial #	6	Student ID	31481
Date	17/8/24	Student Name	G. Naveen

2. Construct a minimum state automaton of the automaton given in Q# 1.

Solution:



3. Differentiate between DFA and 2DFA.

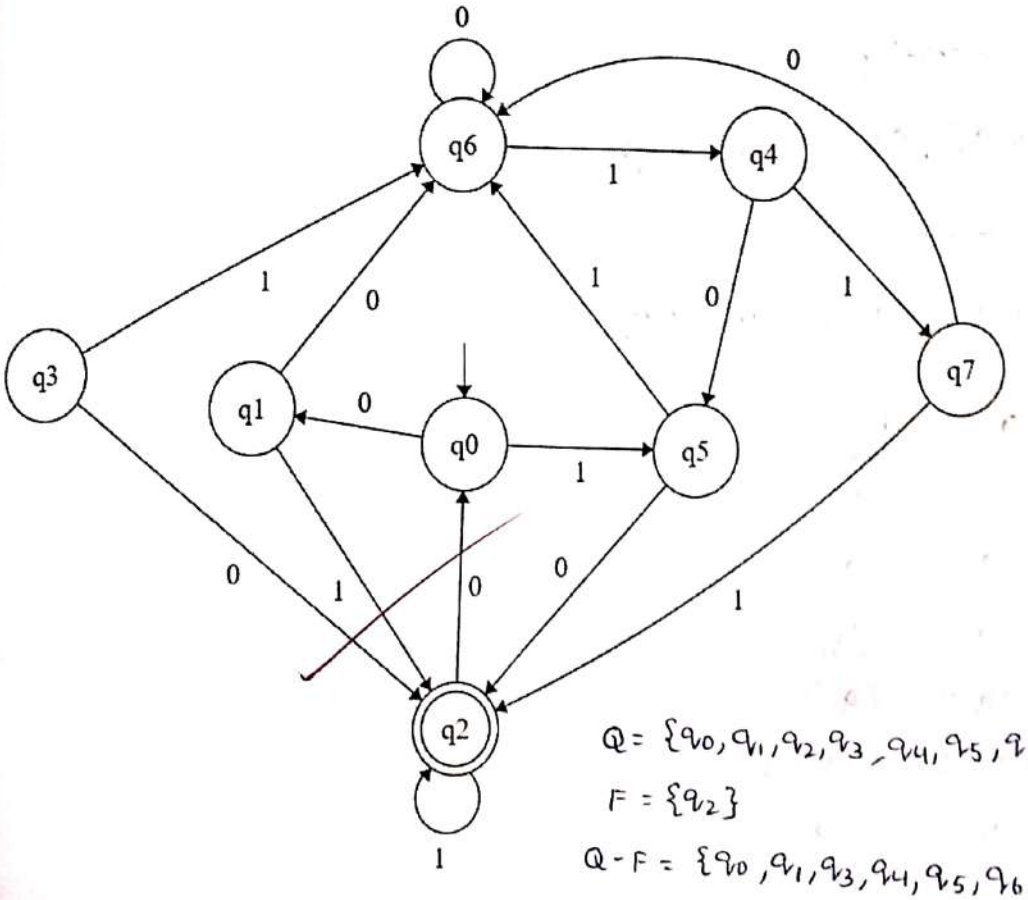
DFA : A DFA processes input strictly left to right with a single pass, recognizing regular languages.

2DFA : It can move its head both left and right, offering more flexibility.

Tutorial #	6	Student ID	31481
Date	17/8/24	Student Name	G. Hanesh

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

.Construct a minimum state automaton equivalent to the finite automaton given below.

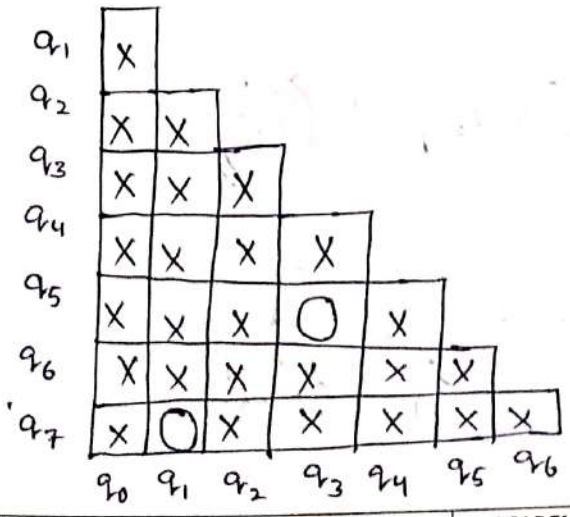


$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$F = \{q_2\}$

$Q - F = \{q_0, q_1, q_3, q_4, q_5, q_6\}$

Solution:



Course Title	THEORY OF COMPUTATION	ACADEMIC YEAR: 2024-25
Course Code(s)	23MT2014	Page 68 of 261

Tutorial #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT> 17/8/24	Student Name	<TO BE FILLED BY STUDENT> G. Vikas Harreesh

After Iteration 1

(q_3, q_0), (q_0, q_4), (q_0, q_5), (q_0, q_6), (q_5, q_3), (q_1, q_7), (q_6, q_4) are empty states

After Iteration 2

(q_7, q_1), (q_5, q_3) are empty

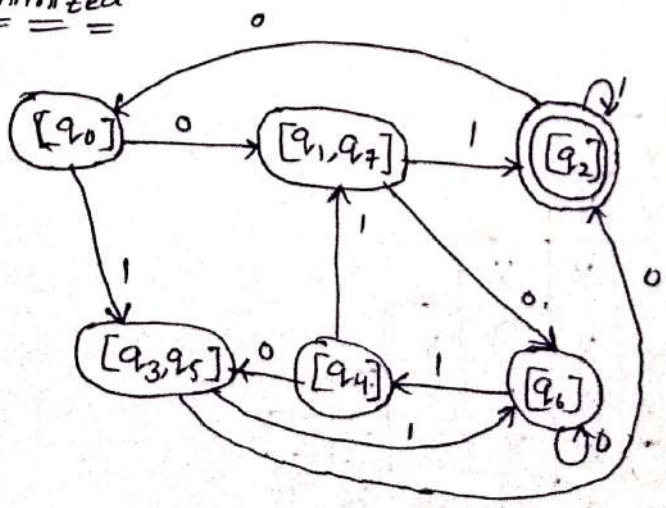
After Iteration 3

(q_7, q_1), (q_5, q_3) are empty

DFA 8'

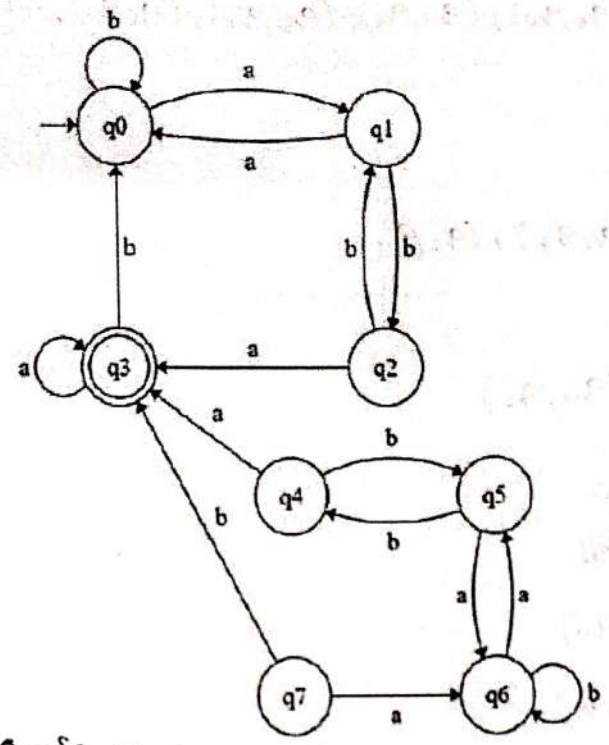
	0	1
→$[q_0]$	$[q_1, q_7]$	$[q_5, q_3]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
*$[q_3]$	$[q_0]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_4]$	$[q_5, q_3]$	$[q_7, q_1]$
$[q_6]$	$[q_6]$	$[q_4]$

DFA minimized



Tutorial #	6	Student ID	3148
Date	17/8/24	Student Name	G. Haneesh

. Construct a minimum state automaton equivalent to the transition diagram given below.

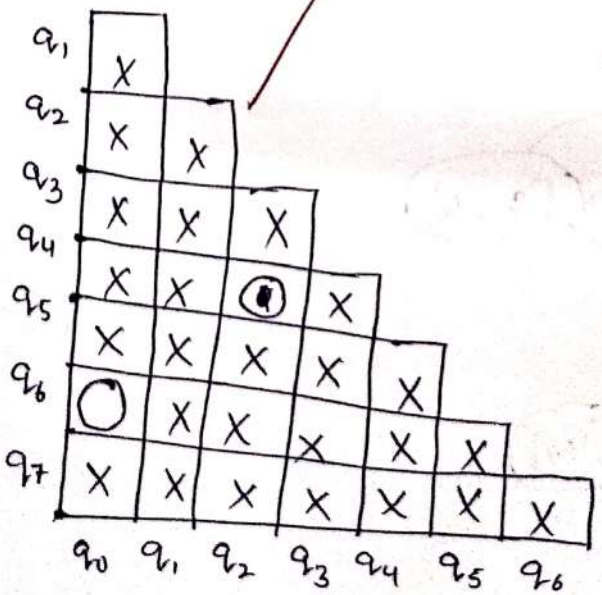


olution:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$

$F = \{q_3\}$

$Q - F = \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}$



Course Title	THEORY OF COMPUTATION	ACADEMIC YEAR: 2024-25
Course Code(s)	23MT2014	Page 70 of 261

Tutorial #	6	Student ID	31481
Date	17/8/24	Student Name	G.Vijay Hari

After iteration 1 (empty state)

$(q_1, q_0), (q_5, q_0), (q_6, q_0), (q_4, q_2), (q_5, q_1), (q_6, q_1), (q_7, q_1)$

After iteration 2

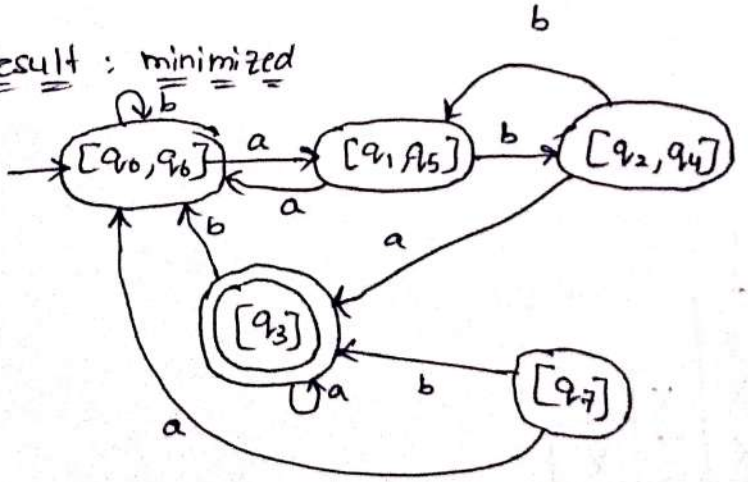
$(q_6, q_0), (q_1, q_5), (q_1, q_7), (q_2, q_4)$

After iteration 3

$(q_6, q_0), (q_5, q_1), (q_4, q_2)$

δ'	a	b
→ $[q_0, q_6]$	$[q_1, q_5]$	$[q_0, q_6]$
$[q_1, q_5]$	$[q_0, q_6]$	$[q_2, q_4]$
$[q_2, q_4]$	$[q_3]$	$[q_1, q_5]$
* $[q_3]$	$[q_3]$	$[q_0, q_6]$
$[q_7]$	$[q_0, q_6]$	$[q_3]$

Result : minimized



Tutorial #	6	Student ID	3481
Date	17/8/24	Student Name	G. Haneesh

Post-Tutorial(To be carried out by student after attending tutorial session)

1. Construct a minimum state automaton equivalent to the given automaton whose transition table is given below. Initial state:q_0, Final State: q_6.

	State	a	b
→	q_0	q_0	q_3
	q_1	q_2	q_5
	q_2	q_3	q_4
	q_3	q_0	q_5
	q_4	q_0	q_6
	q_5	q_1	q_4
*	q_6	q_1	q_3

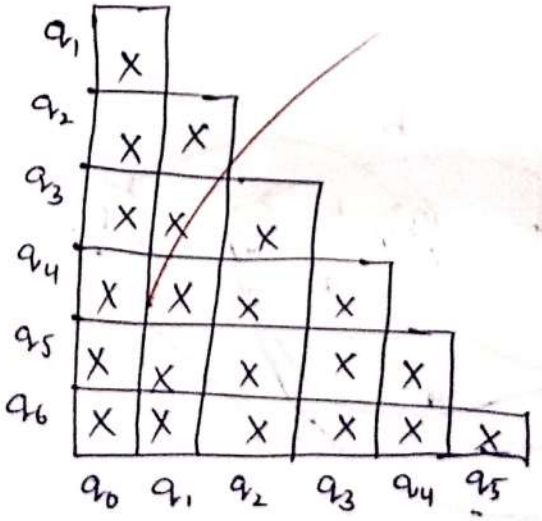
Solution:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$F = \{q_6\}$

$Q - F = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$= \{(q_6, q_0), (q_6, q_1), (q_6, q_2), (q_6, q_3), (q_6, q_4), (q_6, q_5)\}$



Tutorial #	6	Student ID	3148
Date	17/8/24	Student Name	G. Hanee

After Iteration 1 : Empty states are

(q_0, q_1) (q_0, q_5) (q_1, q_5)

(q_0, q_2) (q_1, q_2) (q_2, q_3)

(q_0, q_3) (q_1, q_3) (q_3, q_5)

After Iteration 2 : Empty states are

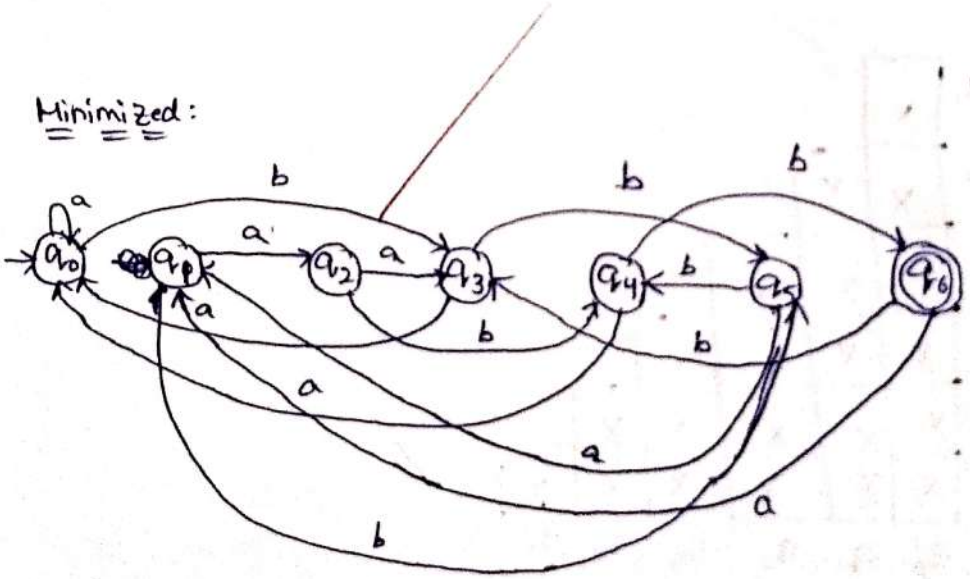
$(q_0, q_1), (q_0, q_3)$

After Iteration 3 :

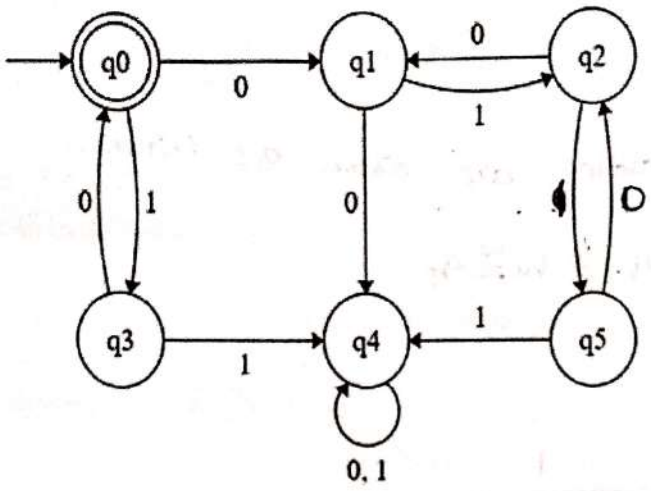
No Empty states

So, the given DFA is minimal DFA

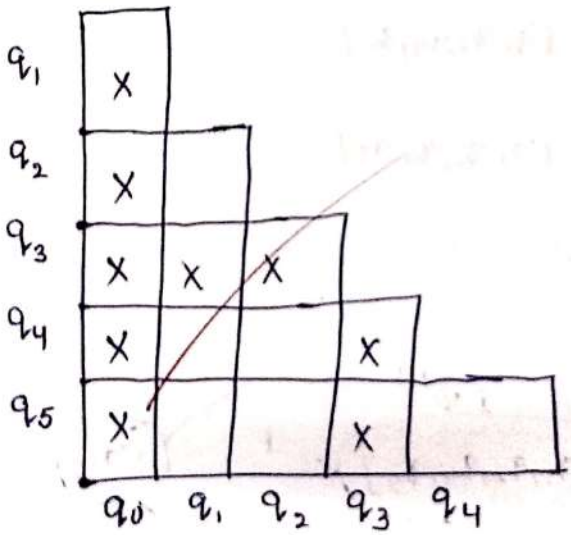
Minimized :



Construct a minimum state automaton equivalent to the DFA described in the following transition diagram.



Solution:



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$F = \{q_0\}$

$Q - F = \{q_1, q_2, q_3, q_4, q_5\}$

Tutorial #	6	Student ID	23181
Date	17/8/24	Student Name	G. Haree

After Heration 1:

pending / Empty states are

$(q_1,q_2), (q_1,q_4), (q_1,q_5), (q_2,q_4), (q_2,q_5), (q_4,q_5)$

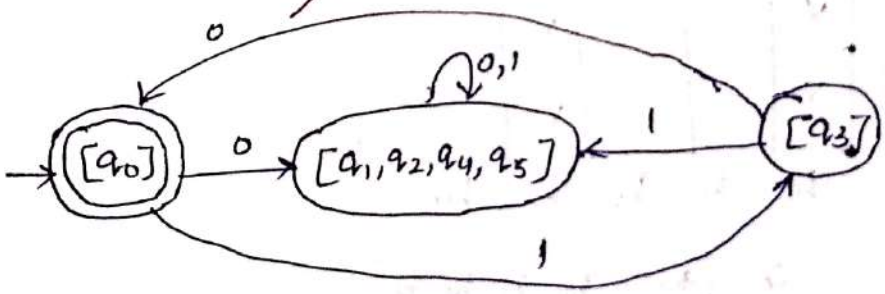
After Heration 2:

pending / Empty states are same as Iteration

$\therefore q_1 \cong q_2 \cong q_4 \cong q_5$

Final DFA

δ'	0	1
* → $[q_0]$	$[q_1,q_2,q_4,q_5]$	$[q_3]$
$[q_1,q_2,q_4,q_5]$	$[q_1,q_2,q_4,q_5]$	$[q_1,q_2,q_4,q_5]$
$[q_3]$	$[q_0]$	$[q_1,q_2,q_4,q_5]$



Tutorial #	<TO BE FILLED BY STUDENT> 6	Student ID	<TO BE FILLED BY STUDENT> 2300031481
Date	<TO BE FILLED BY STUDENT> 17/2/24	Student Name	<TO BE FILLED BY STUDENT> G.V.G.Maneesh

Viva Questions

1. What is the purpose of minimizing a DFA?

Answer:

→ It can be used to reduce the no. of states and transitions of the DFA given.

→ It is used to increase the efficiency of DFA

2. What is the difference between indistinguishable and distinguishable pairs of states in a DFA?

Answer:

Indistinguishable pairs ⇒ Indicates with 'X'
* different states

distinguishable pairs ⇒ Indicates with '0'
* equal states

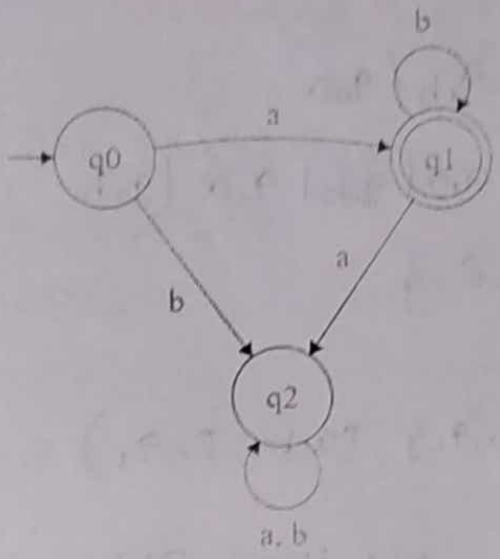
(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: 50 out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: 19/8/2024

Tutorial #	07	Student ID	2300030642
Date	17/09/24	Student Name	S. Nishitha.

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Consider the following automaton that accepts the language ab^*



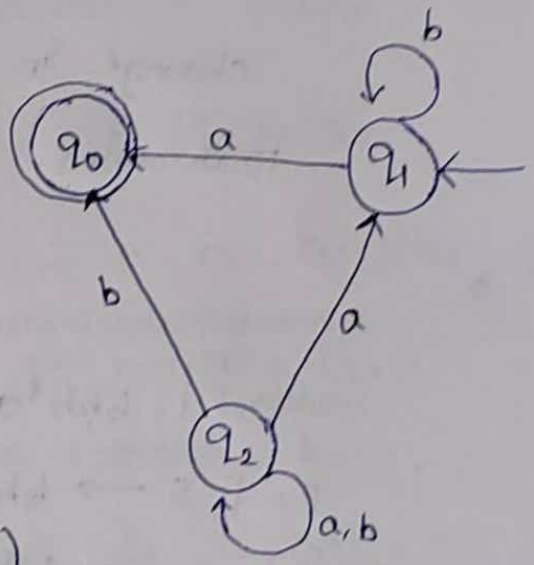
Generate a Right Linear Grammar for the automaton.

Solution:

P: { $q_1 \longrightarrow bq_1 / aq_0$

$q_2 \longrightarrow aq_1 | bq_0 | aq_2 | bq_2$

$q_0 \longrightarrow \in$ }



$G = (\{q_0, q_1, q_2\}, \{a, b\}, P, q_1)$

07.

Date: 17/09/24 Student Name: 2 3000 J064 2 S Nishitha

2. Write the Left Linear Grammar for the above Finite Automaton. What did you ob Explain.

Solution:

$P = \{ q_1 \longrightarrow q_1 b \mid q_0 a$

$q_1 \longrightarrow q_1 a \mid q_2 b \mid q_1 a \mid q_1 b$

$q_0 \longrightarrow \epsilon \}$

$G = (\{q_0, q_1, q_2\}, \{a, b\}, P, q_1)$

Here from right linear grammar, the change in left grammar is only in productions.

3. Write Left-Linear Grammar that generates the language L (bbb^*aaaa^*)

Solution: $L = bbb^*aaaa^*$

$P = \{ S \longrightarrow bbAaaaB$

$A \longrightarrow bA \mid \epsilon$

$B \longrightarrow aB \mid \epsilon \}$

$\Rightarrow$

$\{ S \xrightarrow{aaa} \cancel{A bb}\ \cancel{Baaa}\ bbA$

$A \longrightarrow \cancel{B}bbA \mid \epsilon$

$B \longrightarrow aB \mid \epsilon \}$

Left linear grammar:

$P^1 = \{ S \longrightarrow BaaaAbb$

$A \longrightarrow Abb \mid \epsilon$

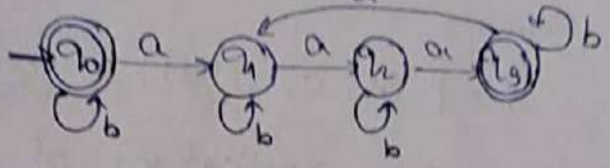
$B \longrightarrow Ba \mid \epsilon \}$

Tutorial No: 07
Date: 17/09/24
Student ID: 23D0030642
Student Name: P. Nishitha

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Find a regular grammar for the language over $\{a, b\}$ where the number of $a's$ in any string is divisible by 3.

Solution: $L = (aaa)^*$



$G = ([q_0, q_1, q_2, q_3], \{a, b\}, P, q_0)$

$P = \{ q_0 \rightarrow bq_0 | aq_1 | \epsilon$

$q_1 \rightarrow bq_1 b | aq_2$

$q_2 \rightarrow bq_2 | aq_3$

$q_3 \rightarrow bq_3 | aq_1 | \epsilon \}$

2. Let L_1 and L_2 be two languages with $L_1 \subseteq L_2$. In what condition L_2 must be a regular language?

Solution: for $L_1 = L_2$ both L_1 and L_2 are regular languages. For $L_1 \subset L_2$, there is no condition for L_2 to be a regular or non-regular Because here there is no condition on L_1. so, $L_1 \subseteq L_2$ is regular and L_2 is also regular language.

3. Let $L = \{w \in \{0,1\}^* : n_0(w) = n_1(w)\}$. Is L regular? If you find it to be not regular, prove it. Here $n_0(w)$ = count of zeros in w, and $n_1(w)$ = count of 1's in w

Solution:

$L = \{01, \epsilon, 0101, 1100, 001011, \ldots\}$

By using applications of pumping lemma,

Step1:

let, constant $n = 4$

Step2: $z = uvw$ $|z| \geq n$

where, $|uv| \leq n$

$|v| \geq 1$

$z = 1100$ (u = 1, v = 1, w = 00)

$4 \geq 4$

$|uv| \leq n \Rightarrow 2 \leq 4$ ✓

$|v| \geq 1 \Rightarrow 1 \geq 1$ ✓

$uv^iw \in L$

$i = 0, 1, 2, 3, \ldots$

$i = 0 \rightarrow 1(1)^0 00 = 100 \notin L$

L is not regular.

Tutorial No. 07 Student ID 2100030642
Date 12/9/24 Student Name [illegible]

4. Is the language $L = \{a^n b^n : n \geq 0\}$ a regular language? If you find it to be not a regular language prove it. Write each step.

Solution:

$L = \{ \epsilon, ab, aabb, aaabbb, \cdots \}$

1. let $n = 4$

2. $z = uvw$

$z = \underbrace{aa}_{u}\underbrace{ab}_{v}\underbrace{bb}_{w}$

$|z| \geq n \Rightarrow 4 \geq 4$

$|uv| \leq n \Rightarrow 2 \leq 4$

$|v| \geq 1 \Rightarrow 1 \geq 1$

3. $uv^i w$

$i = 0, 1, 2, 3, \cdots$

$i = 0 \Rightarrow a(a)^0 bb = abb \notin L$

$\therefore$ L is not regular.

5. Consider the grammar: $S \rightarrow aS \mid bA$, $A \rightarrow bA \mid \varepsilon$. Is it a regular gram... it to be a regular grammar determine whether it is right linear or left li... language generated by the given grammar.

Solution:

$P = \{ S \rightarrow aS \mid bA$

$A \rightarrow bA \mid \epsilon \}$

$G = (\{S, A\}, \{a, b\}, P, S)$

It is regular grammar.

Because it consists,

it is in the form T^*V / T^*.

Given production is in the form of

T^*V

so, it is Right regular grammar.

Date 10/9/24 Student Name

Post-Tutorial (To be carried out by student after attending tutorial session)

Consider the grammar: $S \rightarrow aS \mid bS \mid \varepsilon$. Is it a regular grammar? If you find it to be regular, determine its type. Write the language generated by the grammar.

ution:

It is in the form of $T^*V \mid T^*$. So, it is regular grammar.

$S \rightarrow aS \mid bS \mid \epsilon$

$\therefore L = (a^*b^*)^*$

Let $L_1 = \{a^n : n > 0\}$ and $L_2 = \{b^n : n > 0\}$ be two regular languages. Consider another language L_3 where $L_3 = L_1 \cup L_2$. Is L_3 a regular language? Explain.

olution:

If L_1 is a regular language.

L_2 is a regular language.

If L_1 and L_2 are regular languages, then $L_3 = L_1 \cup L_2$ is regular language.

$L_3 = \{a^n, b^n ; n > 0\}$

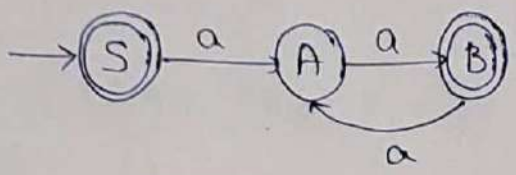
$\therefore L_3$ is regular language.

Tutorial #	07	Student ID	2100310642
Date	17/9/24	Student Name	[illegible]

3. Consider the language $L = \{a^n : n \text{ is even}\}$. Is L a regular language? P

Solution:

$L = \{\epsilon, aa, aaaa, \cdots\} = (aa)^*$



$P = \{ S \rightarrow aA \mid \epsilon$

$A \rightarrow aB$

$B \rightarrow aA \mid \epsilon \}$

$G = (\{S, A, B\}, \{a\}, P, S)$

∵ It is regular language $(T^*V|T^*)$

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: _____ out of 50
	Full Name of the Evaluator:
	Signature of the Evaluator Date of
	Evaluation:

Tutorial No. 05 Student ID 2300030642
Date 17/9/24 Student Name Nikhil

Pre-Tutorial (To be completed by student before attending tutorial session)

1. What is the language generated by the grammar? $G = \{S \rightarrow aSb \mid SS \mid \varepsilon\}$

Solution: $S \rightarrow asb \mid ss \mid \epsilon$

$asb \Rightarrow ab$

asb

$aasbb \Rightarrow aabb$

$assb$

$aasbasbb$

$\rightarrow aababb$

language of the given grammar is the combination of a's and b's.

2. Given the production rules of a simple CFG, Use the following grammar to derive the string from the start symbol?

Non-terminals: {S,A}, Terminals: {a,b}, Start Symbol: S

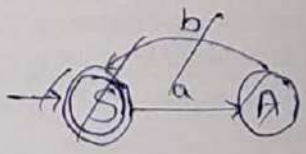
Production Rules:

a. S→aA
b. A→ bS
c. S→ϵ

Solution:

$S \rightarrow aA \mid \epsilon$

$A \rightarrow bS$



$S \rightarrow aA$

abS

$\rightarrow ab$.

It is not possible to generate 'abb' string from given grammar.

Tutorial #	05	Student ID	2300020642
Date	17/9/24	Student Name	Nishitha

3. How does a parse tree represent the derivation of a string in a CFG, and why is it useful in understanding the structure of the generated strings? Can you draw a parse tree for the string "(())" using the CFG for balanced parentheses:

Non-terminals: {S}, Terminals: {(,)}, Start Symbol: S

Production Rules:

a. S→ SS

b. S→(S)

c. S→ϵ

Solution:

S → (S)

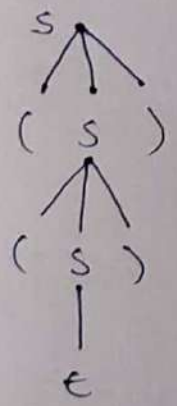
→ ((S))

→ "(())"

→ (S) ⟹ S replace with (S)

((S)) → S replace with epsilon

"(())"



→ parse tree

∴ "(())"

Tutorial # 01 Student ID 2300030642
Date 17/9/24 Student Name Nelluri

4. What is the difference between leftmost derivation and rightmost derivation in the co
CFGs? Provide an example of each for the string "ab" using the following grammar:

Non-terminals: {S}, Terminals: {a, b}, Start Symbol: S
Production Rules:

c. S→aS
d. S→bS

Solution:

S→aS

S→bS

Here, variable on the right most. So, for given grammar there is no variable that leftmost.

So, we can implement right most derivation.

S → aS
aBS
abas, abbs

If S → ∈ then

→ abS
"ab".

Tutorial # 05 Student ID 230080642
Date 17/9/24 Student Name T. Nischitha.

5. Is the following grammar in CNF?

$$S \rightarrow aS \mid Sb \mid AB$$

$$A \rightarrow a \mid \varepsilon$$

$$B \rightarrow b$$

If you find it to be not in CNF, convert it into CNF. Write each step.

Solution:

Step1: After simplification of given grammar.

$P' = \{ S \rightarrow aS \mid Sb \mid b \mid AB$

$A \rightarrow a$

$B \rightarrow b \}$

Step2: already CNF $\Rightarrow$ $S \rightarrow b$

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

Introduce two new variables C_a & C_b. $C_a \rightarrow a$, $C_b \rightarrow b$

$S \rightarrow aS$ $\quad S \rightarrow Sb$

$C_aS \quad \rightarrow SC_b$

$G' = (\{S, A, B, C_a, C_b\}, \{a, b\}, P^*, S)$

$P = \{ S \rightarrow b \mid AB \mid C_aS \mid SC_b$

$A \rightarrow a$

$B \rightarrow b$

$C_a \rightarrow a$

$C_b \rightarrow b \}$

Tutorial # 05 Student ID 2100030642
Date 17/9/24 Student Name Nikitha

6.Consider the language $L = \{a^n b^m : n \neq 2m\}$. Is it a regular, context-free or both regular, context-free language? Prove it.

Solution:

L = { ab, abbb, ···· , aab, aabb, aabbb, ···· }

Check regular:

$z = uvw = aabbb$

i=0 → abb ∈L i=2 → aaabb ∈L

i=1 → aabb ∈L

∴ It is not a regular.

If some condition it possible for $n = 2m$. So, it is not regular by using pumping lemma.

It is a context free grammar because it is derived from a grammar in the form $(V \cup T)^*$.

∴ It is not regular

It is context free grammar.

Tutorial # 05 Student ID 230003642
Date 27/9/24 Student Name Nirchita

7. Consider the following grammar

$$S \rightarrow aSa \mid bSb \mid \varepsilon$$

Is it in Griebach normal form? If not convert it into Greibach normal form.

Solution: After simplification of given grammar,

Step 1:

$S \rightarrow aSa \mid aa \mid bSb \mid bb$

Step 2: (Somewhat CNF)

$S \rightarrow aSa$

$S \rightarrow aa$

$S \rightarrow bSb$

$S \rightarrow bb$

Introduce two new variables C_a, C_b, $C_a \rightarrow a$, $C_b \rightarrow b$

$S \rightarrow aSC_a \mid aC_a$

$S \rightarrow bSC_b \mid bC_b$

$P' = \{ S \rightarrow aSC_a \mid aC_a \mid bSC_b \mid bC_b \}$

$G' = (\{S, C_a, C_b\}, \{a, b\}, P', S)$

Tutorial #: 05 Student ID: 2300030642
Date: 19/9/24 Student Name: Alisha.

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Consider the following grammar

$$S \rightarrow ASB \mid c$$

$$A \rightarrow \epsilon \mid aA$$

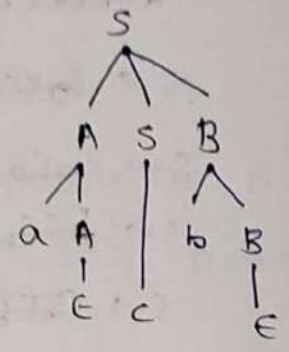
$$B \rightarrow \epsilon \mid bB$$

Derive the string acb using leftmost and rightmost derivation. Show parse trees for your derivation.

Solution:

$S \rightarrow ASB$ (leftmost derivation)

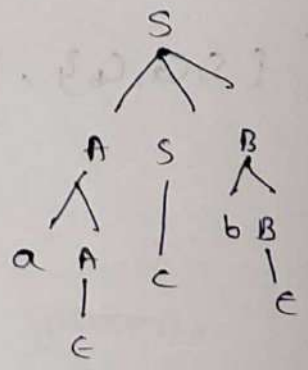
$S \rightarrow aASB$
$\rightarrow aSB$
$\rightarrow acB$
$\rightarrow acbB$
$\rightarrow 'acb'$



acb

(Right most derivation)

$S \rightarrow ASB$
$\rightarrow ASbB$
$\rightarrow Asb$
$\rightarrow Acb$
$\rightarrow aAcb$
$\rightarrow 'acb'$



05
17/9/24

2300010642
Nikhil C.

ven the language consisting of strings with equal numbers of a's and b's (e.g., 'ab', 'aabb', 'abab').
truct a CFG that generates this language. What challenges do you face in ensuring that the numbers of a's
b's are equal?

tion: $L = \{ ab, aabb, abba, abab, abbbaa, \ldots \}$

from given language the base case is 'ϵ'

$$S \rightarrow \epsilon$$

for 'ab':

$$S \rightarrow asb$$

for 'ba':

$$S \rightarrow bSa$$

for mixing of same number of a's and b's:

$$S \rightarrow SS$$

So, the CFG is

$P = \{ S \rightarrow \epsilon \mid asb \mid bsa \mid SS \}$

Experiment #
Date

Student ID
Student Name

3. Draw the parse tree for the string "aabbb" using the following CFG:
Non-terminals: (S,A), Terminals: (a, b), Start Symbol: S
Production Rules:

i) S→aSb
ii) S→A
iii) A→bA
iv) A→ε

Solution:

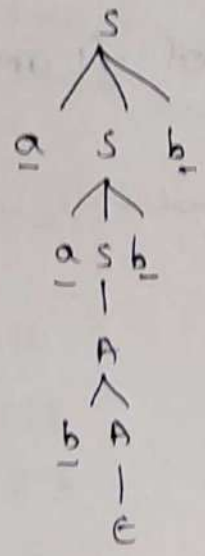
$S \rightarrow aSb$

$\rightarrow aaSbb$ ($\because S \rightarrow aSb$)

$\rightarrow aaAbb$ ($\because S \rightarrow A$)

$\rightarrow aabAbb$ ($\because A \rightarrow bA$)

$\rightarrow$ "aabbb" ($\because A \rightarrow \epsilon$)



"aabbb".

Exp. No. : 05 Student ID : 2100080642
Date : 19/9/24 Student Name : Nikhila

1. Using the following CFG, perform both a leftmost and a rightmost derivation for the string "aab":
Non-terminals: {S,A}, Terminals: {a, b}, Start Symbol: S,
Production Rules:
S→aS, S→A, A→aA, A→b

Solution:

$S \rightarrow aS$

$S \rightarrow A$

$A \rightarrow aA$

$A \rightarrow b$

Left most derivation:

$S \rightarrow aS$

$\rightarrow aA$

$\rightarrow aaA$

$\rightarrow$ "aab"

Right most-derivation:

$S \rightarrow aS$

$\rightarrow aA$

$\rightarrow aaA$

$\rightarrow$ "aab".

Experiment	01	Student ID	2300030642
Date	17/9/24	Student Name	Mirlade

5.Consider the CFG for simple arithmetic expressions involving addition and multiplication:

Non-terminals: {E, T, F}, Terminals: {+, *, (,), id}, Start Symbol: E

Production Rules:

E→E+T, E→T, T→T*F, T→F, F→(E), F→id

Parse the expression "id + id * id" and draw the corresponding parse tree. Explain each step in t process and how the CFG handles operator precedence.

Solution:

$E \rightarrow E+T$

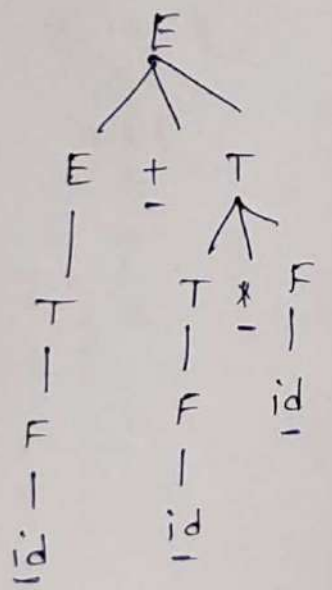
$\rightarrow E+T*F$

$\rightarrow E+F*F$

$\rightarrow T+F*F$

$\rightarrow F+F*F$

$\rightarrow$ "id + id * id"



"id + id * id"

Experiment # 5 Student ID 2100010642
Date 17/9/24 Student Name S. Nikhil Re.

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Consider the grammar G = (V, Σ, R, S), where V = {a, b, S, A}, Σ = {a, b}, R = { S → AA, A → AAA, A → a, A → bA, A → Ab }.
   (1) List the set of strings that can be produced by derivations of four or fewer steps using G
   (2) (b) Give any four distinct derivations for the string "babbab" using G

Solution:

(1) $S \to AA \to aA \to "aa"$

$S \to AA \to bAA \to bbAA \to "bbaa"$

$S \to bbAA \to bbbaa$

$S \to AA \to AbAb \to "abab"$

(2) $S \to AA \to bAA \to baA \to babA \to babbA \to babbAb \to "babbab"$

$S \to AA \to AAA \to bAAA \to baAA \to babAA \to babbAA \to "babbab"$

(1) : "aa", bbaa, bbbaa, abab

2. Design CFG for a language which accepts palindrome over an alphabet Σ = {a,b}

Solution:

$L = \{ aa, aba, bab, \cdots \}$

for empty:

$S \to \epsilon$

for aa

$S \to aSa$

for aba:

$S \to b$

for bab

$S \to bSb \mid a$

for mixing : $S \to SS$

$\therefore P' = \{ S \to \epsilon \mid aSa \mid bSb \mid a \mid b \mid SS \}$

Experiment #	05	Student ID	21000...
Date	17/9/24	Student Name	S.Nishi...

3. Check whether the given grammar is ambiguous or not: S → SS, S → a, S → b. Expla...

Solution:

S → SS

S → a

S → b

leftmost:

S → SS

→ aS

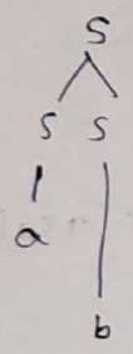
→ "ab"

Rightmost:

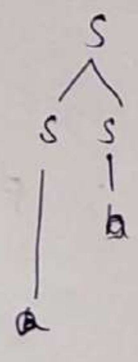
S → SS

→ Sa

→ "ba"



"ab"



"ba"

∵ It is ambiguous.

Date	12/9/24	Student ID	23000306 42
		Student Name	Nishitha

Viva Questions

1. What is the role of non-terminal and terminal symbols in a CFG?

Answer:

Terminals symbols are alphabets, digits, special symbols

Eg: a, b, (,), *, 2, 1, ---

Non-terminals means variables (uppercase letters)

Eg: A, S, B, ---

2. Provide an example of a production rule and explain its significance.

Answer:

Production is the form of $\alpha \rightarrow \beta$

where

$\alpha \in V$

$\beta \in V^* | T^* | (VUT)^*$

Eg: i) $A \rightarrow aSa$

ii) $B \rightarrow b$

iii) $S \rightarrow SA$

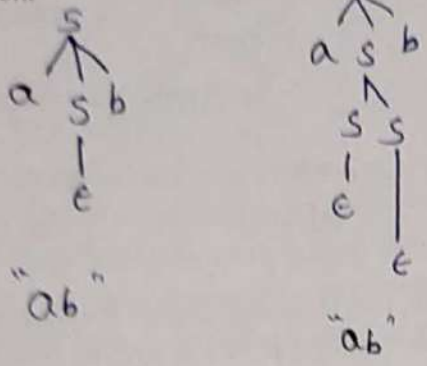
(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: out of 50
	Full Name of the Evaluator:
	Signature of the Evaluator Date of Evaluation:

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Is G:{S → SS | aSb | bSa | ε } ambiguous? Explain your answer.

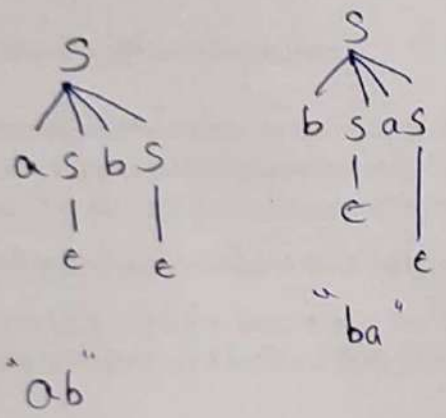
Solution:



∴ It is ambiguous

2. Is G: {S → aSbS | bSaS| ε } ambiguous? Explain your answer.

Solution:



∴ It is not ambiguous

Student ID: 30642
Student Name: Nithila.

17/9/24

Consider the following grammar

$S \rightarrow aS \mid \epsilon$

The language generated by this grammar is $L = \{a^n : n \geq 0\}$ or a^*

a. Find the Leftmost Derivation and Rightmost Derivation.

b. Prove All the strings generated the above grammar have their leftmost derivation and rightmost derivation exactly same. Draw the Parse tree for the same.

Solution:

a. Leftmost:

$S \rightarrow aS$
$\rightarrow a\epsilon$
$\rightarrow$ "a"

$S \rightarrow aS$
$\rightarrow aaS$
$\rightarrow$ "aa"

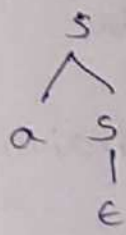
Rightmost:

$S \rightarrow aS$
$\rightarrow a\epsilon$
$\rightarrow$ "a"

$S \rightarrow aS$
$\rightarrow aaS$
$\rightarrow$ "aa"

b. Leftmost

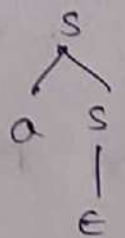
"a" $\rightarrow$



"a"

Rightmost

"a" $\rightarrow$



"a"

∴ Leftmost derivation = Rightmost derivation.

Experiment 9 Student ID 3064L

Date 17/9/24 Student Name Nishita.

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Construct a CFG for a language $L = \{wcwR \mid w \in (a, b)^*\}$.

Solution: P: { S → aSa | bSb | c | ∈ | aSb | bSa }

wcw^R ⇒ "w" may a or b

If "a", aca^R

If "b", bcb^R

S → c | aSa | bSb

S → ∈ (if no a's or b's)

2. Derive the string "*aabbabba*" for left most derivation and right most derivation using above CFG.

Solution:

S → aSa

→ aaSba

→ aabSbba

→ aabbSabba

→ "aabbabba"

S → aSa

→ aaSba

→ aabSbba

→ aabbSabba

→ "aabbabba"

Both leftmost and Rightmost derivations are equal.

9
17/9/24

30642
Nichita

3. Consider the following grammar

$$S \rightarrow ASB \mid c$$

$$A \rightarrow \epsilon \mid aA$$

$$B \rightarrow \epsilon \mid bB$$

Derive the string *acb* using leftmost and rightmost derivation. Show the parse trees for your derivation.

Solution:

leftmost:

$S \rightarrow ASB$

$\rightarrow aASB$

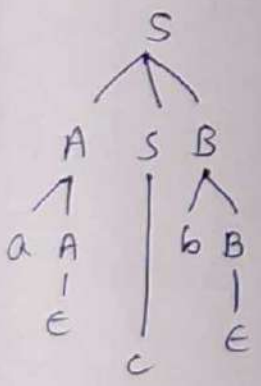
$\rightarrow a\epsilon SB$

$\rightarrow aSB$

$\rightarrow acB$

$\rightarrow acbB$

$\rightarrow$ "acb"



"acb"

Rightmost:

$S \rightarrow ASB$

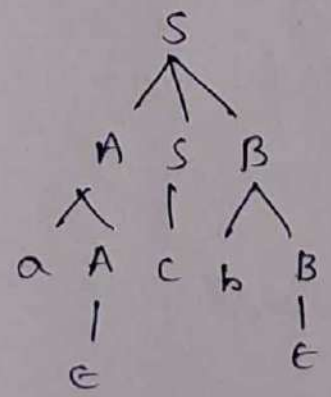
$\rightarrow ASbB$

$\rightarrow ASb$

$\rightarrow Acb$

$\rightarrow aAcb$

$\rightarrow$ "acb"



"acb"

Experiment # 9 Student ID 30642

Date 12/9/24 Student Name Nikhita

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Generate CFG for the language $L = \{0^i 1^j 0^k : j > i + k\}$

Solution:

for i=0, j=2, k=0 ("11")

$S \rightarrow A1B$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

for i=1, j=3, k=1 ("01110")

$S \rightarrow A1B$

$A \rightarrow 0$

$B \rightarrow 0$

for i=2, j=4, k=2 ("00111100")

$S \rightarrow A1B$

$A \rightarrow 00$

$B \rightarrow 00$

CFG:

$S \rightarrow A1B$

$A \rightarrow 0A \mid 0 \mid \epsilon$

$B \rightarrow 0B \mid 0 \mid \epsilon$

9
17/9/24

30642
Nithila

2. Consider the following grammar-

$$S \rightarrow SaS \mid b$$

Is it an ambiguous grammar? Generate the string *babab* from this grammar to prove your point.

Solution:

$S \rightarrow SaS$

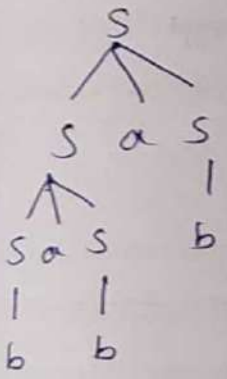
$\rightarrow$ SaSaS

$\rightarrow$ baSaS

$\rightarrow$ babaS

$\rightarrow$ "babab"

Leftmost :



"babab"

$S \rightarrow SaS$

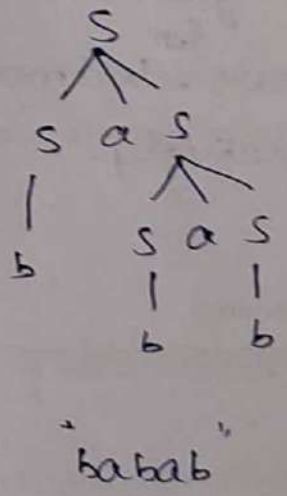
$\rightarrow$ SaSaS

$\rightarrow$ SaSab

$\rightarrow$ Sabab

$\rightarrow$ "babab"

Rightmost :



"babab"

∴ It is ambiguous.

Experiment #	9	Student ID	30642
Date	12/9/24	Student Name	Nishita

Viva Questions

1. What is a context-free grammar (CFG), and how is it defined?

Answer:

It is 4-tuple representation $G = (V, T, P, S)$

$P : \alpha \rightarrow \beta$

$\alpha \in V$

$\beta \in (V \cup T)^*$

2. How do you construct a parse tree for a given string using a context-free grammar?

Answer:

By substituting the terminals to generate a string.

In that process we form leftmost and Rightmost derivations by substituting terminals from leftmost and from rightmost respectively.

(For Evaluator's use only)

	Comment of the Evaluator (if Any)	Evaluator's Observation
		Marks Secured: out of 50
		Full Name of the Evaluator:
		Signature of the Evaluator Date of Evaluation:

Experiment # 10 Student ID 30642

Date 19/4/24 Student Name Nikhila

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Remove *null* productions from the following CFG

$$S \rightarrow ASA \mid aB \mid \epsilon, \quad A \rightarrow B \mid S \mid \epsilon, \quad B \rightarrow b \mid \epsilon$$

Solution:

$S \rightarrow ASA \mid aB \mid \epsilon$

$A \rightarrow B \mid S \mid \epsilon$

$B \rightarrow b \mid \epsilon$

Nullable :

$S \rightarrow \epsilon$

$A \rightarrow \epsilon$ } Eliminate.

$B \rightarrow \epsilon$

$S \rightarrow ASA$ (i) $S \rightarrow SA$

$S \rightarrow AA$

$S \rightarrow AS$

$S \rightarrow A$

$S \rightarrow S$

$S \rightarrow aB$

$\Downarrow$

$S \rightarrow a$

After.

$\therefore P = \{S \rightarrow ASA \mid aB \mid SA \mid AA \mid AS \mid A \mid S \mid a$

$A \rightarrow B \mid S$

$B \rightarrow b\}$

Experiment #
Date 19/9/24

Student ID 30642
Student Name Nirmit Ca

2. Remove *unit* productions from the following.

$S \to A, \quad A \to B, \quad B \to C \mid a, \quad C \to b \mid c$

Solution:

$S \to A$

$A \to B$

$B \to C \mid a$

$C \to b \mid c$

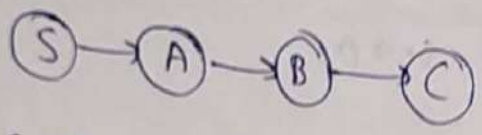
Unit productions:

$S \to A$

$A \to B$

$B \to C$

Dependency graph:



$S \to B$

$A \to C$

(i) $S \to A$
$\to B$
$\to a$
$S \to a$

(ii) $A \to B$
$A \to a$
$A \to C$
$A \to b$
$A \to c$

(iii) $B \to C$
$B \to b$
$B \to c$

(iv) $S \to B$
$S \to a$
$\to C$
$S \to b$
$S \to c$

(v) $A \to C$
$A \to b$
$A \to c$

$\therefore P = \{ S \to a \mid b \mid c$

$A \to a \mid b \mid c$

$B \to a \mid b \mid c$

$C \to b \mid c \}$

10
11/9/24

30642
Nishita.

3.Simplify the following grammar:

$$S \rightarrow Aa \mid B, \quad B \rightarrow a \mid bC, \quad C \rightarrow a \mid \varepsilon$$

Mention the number of productions in your simplified grammar.

Solution:

(i) ϵ-productions :

$C \rightarrow \epsilon$ (eliminate)

$B \rightarrow bC$

$B \rightarrow b$

$S \rightarrow Aa \mid B$

$B \rightarrow a \mid bC \mid b$

$C \rightarrow a$

(ii) unit productions :

$S \rightarrow B$ (eliminate)

$S \rightarrow a$

$S \rightarrow B$

$\rightarrow bC$

$S \rightarrow b$

$S \rightarrow ba$

$S \rightarrow Aa \mid B \mid a \mid b \mid ba$

$B \rightarrow a \mid bC \mid b$

$C \rightarrow a$

(iii) Useless symbols:

$S \rightarrow Aa$

useless : {A}

$S \rightarrow B$

$S \rightarrow "a"$

$\{S \rightarrow B \mid a \mid b \mid ba$

$B \rightarrow a \mid bC \mid b$

$C \rightarrow a$ }

∴ No. of productions : 8

Date 18/9/24 Student Name 30641 Nikhil Ca

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1.Simplify the following CFG.

$S \rightarrow AB \mid AC, \quad A \rightarrow aB \mid BC \mid \epsilon, \quad B \rightarrow b, \quad C \rightarrow c$

Solution:

(i) ϵ-productions

$A \rightarrow \epsilon$ (eliminate)

$S \rightarrow AB$ $\quad S \rightarrow AC$

$\boxed{S \rightarrow B}$ $\quad \boxed{S \rightarrow C}$

$S \rightarrow AB \mid AC \mid B \mid C$

$A \rightarrow aB \mid BC \mid B$

$B \rightarrow b$

$C \rightarrow c$

(ii) Unit productions : NIL

(iii) Useless symbols

1) $S \rightarrow AB$ $\quad S \rightarrow AC$ $\quad A \rightarrow aB$ $\quad A \rightarrow BC$

$\rightarrow aBb$ $\quad \rightarrow aBc$ $\quad \rightarrow$ "ab" $\quad \rightarrow$ "bc"

$\rightarrow$ "abb" $\quad \rightarrow$ "abc"

2) $S \rightarrow AB$, $\quad S \rightarrow AC$

aBb $\quad \rightarrow aBc$

"abb" $\quad \rightarrow$ "abc"

NO useless symbols.

$\therefore \{ S \rightarrow AB \mid AC \mid B \mid C$

$A \rightarrow aB \mid BC$

$B \rightarrow b$

$C \rightarrow c \}$

Experiment # AD
Date 18/9/24
Student ID 30642
Student Name [illegible]

Simplify the following CFG.

$S \rightarrow AB \mid \epsilon, \quad A \rightarrow aB \mid \epsilon, \quad B \rightarrow bA \mid b, \quad C \rightarrow bC$

Solution:

(i) ϵ - productions

$S \rightarrow \epsilon$ & (eliminate)
$A \rightarrow \epsilon$

$S \rightarrow AB \quad B \rightarrow bA$

$\boxed{S \rightarrow B} \quad \boxed{B \rightarrow b}$

$S \rightarrow AB \mid B$
$A \rightarrow aB$
$B \rightarrow bA \mid b$
$C \rightarrow bC$

(ii) unit productions : NIL

(iii) useless symbols:

$C \rightarrow bC$
$\rightarrow$ (x)

useless : {C}

CFG:

$P = \{ S \rightarrow AB \mid B$
$A \rightarrow aB$
$B \rightarrow bA \mid b \}$

Experiment # 10 Student ID 20642

Date 11/9/24 Student Name Nikhil

3. Simplify the following CFG.

$S \rightarrow ABCD \mid AC, \quad A \rightarrow aA \mid \epsilon, \quad B \rightarrow bB \mid bC \mid b, \quad C \rightarrow cC \mid D,$

Solution:

(i) ϵ Production

$A \rightarrow \epsilon$ (eliminate)

$S \rightarrow ABCD \quad S \rightarrow AC \quad A \rightarrow aA$

$\boxed{S \rightarrow BCD} \quad \boxed{S \rightarrow C} \quad \boxed{A \rightarrow a}$

(ii) Unit Productions:

$C \rightarrow D$

$\boxed{C \rightarrow d}$

(iii) useless symbols:

1) $S \rightarrow ABCD \quad S \rightarrow AC \quad A \rightarrow aA \quad B \rightarrow bB$

$aABCD \quad S \rightarrow (X) \quad (X) \quad \rightarrow bb'$

useless: {A}

$B \rightarrow bC$

$B \rightarrow bDE$

$\rightarrow bd'$

2) $S \rightarrow ABCD \quad S \rightarrow AC$

$(X) \quad (X)$

CFG: $G = (\{S, A, B, C, D\}, \{a, b, c\}, P, S)$

$P = \{ S \rightarrow BCD \mid C \quad C \rightarrow d \mid D \mid cC$

$A \rightarrow a \quad D \rightarrow dy$

$B \rightarrow bB \mid bC \mid b$

Experiment 8 | Student ID: 30642
Date: 10/9/24 | Student Name: Nishita

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Simplify the following CFG : $S \rightarrow AB,\ A \rightarrow aA \mid \epsilon,\ B \rightarrow bB \mid b$

Solution:

(i) ϵ-productions

$A \rightarrow \epsilon$ (elim)

$S \rightarrow AB$ $\quad$ $A \rightarrow aA$

$S \rightarrow B$ $\quad$ $A \rightarrow a$

(ii) unit productions : NIL

(iii) useless symbols:

1) $S \rightarrow AB$ $\quad$ $A \rightarrow aA$ $\quad$ $B \rightarrow bB$

$\rightarrow aAB$ $\quad$ ✗ $\quad$ $\rightarrow$ "b"

$\rightarrow aAb$

✗

useless: {A}

2) $S \rightarrow AB$

✗

$G = (\{S, A, B\}, \{a, b\}, P, S)$

$P : \{ S \rightarrow B$

$A \rightarrow a$

$B \rightarrow bB \mid b \}$

Experiment # 1P Student Name Nishitha

Date 15/9/24

2. Remove useless symbols from the following CFG

$$T \rightarrow xxY \mid xbX \mid xxT, \quad X \rightarrow xX, \quad Y \rightarrow xy \mid y, \quad Z \rightarrow xz$$

Solution:

1) $T \rightarrow xxY$ $T \rightarrow xbX$ $T \rightarrow xxT$

$\rightarrow$ "xxy" $\rightarrow$ xbX $\rightarrow$ xxxxY

$\rightarrow$ xbxX $\rightarrow$ "xxxxy"

(x)

useless: {X}

$X \rightarrow xX$ $Y \rightarrow$ "xy" $Z \rightarrow$ "xz"

(x) $Y \rightarrow$ "y"

2) $T \rightarrow xxY$ $T \rightarrow xbX$ $T \rightarrow xxT$

"xxy" (x) "xxxxy"

useless: {X}

$\therefore \{T \rightarrow xxY \mid xxT$

$Y \rightarrow xy \mid y$

$Z \rightarrow xz\}$

ent #	10	Student ID	30642
	1Malu	Student Name	Nishitha.

emove null production from the following

$$S \rightarrow ABA, \quad A \rightarrow 0A \mid \varepsilon, \quad B \rightarrow 1B \mid \varepsilon$$

tion:

$A \rightarrow \epsilon$
$B \rightarrow \epsilon$ } eliminate

$S \rightarrow ABA$ (i) $S \rightarrow BA$

$S \rightarrow AA$

$S \rightarrow AB$

$S \rightarrow A$

$S \rightarrow B$

$A \rightarrow 0A$ $\quad$ $B \rightarrow 1B$

$A \rightarrow 0$ $\quad$ $B \rightarrow 1$

$\therefore$ $S \rightarrow ABA/BA/AA/AB/A/B$

$A \rightarrow 0A/0$

$B \rightarrow 1B/1$

Experiment #	10	Student ID	30642
Date	11/9/24	Student Name	Nishac

Viva Questions

1. Why is it important to simplify context-free grammars?

Answer:

To construct Chomsky normal form and Gray Normal form, we need to simplify the CFG to convert easily with ε, unit and useless productions.

3. Does simplifying a context-free grammar change the language it generates? Why or why not?

Answer:

By simplifying CFG the language doesnot changes. Because by simplifying CFG, even we eliminate the ε-productions, unit productions and useless symbols, the language can't change only the productions will be change.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment # 11 | Student ID 70642
Date 11/9/24 | Student Name Nikhita

Introduce New Non-terminals: For terminal symbols in longer productions, introduce new symbols to ensure all rules fit the binary or terminal production form.

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Remove unit productions from the following grammar

$$S \to AC,\ A \to a,\ C \to X | b,\ X \to Y,\ Y \to Z,\ Z \to a$$

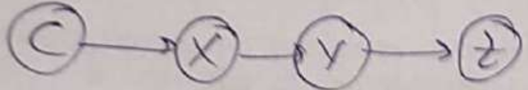
Solution:

$C \to X$

$X \to Y$

$Y \to Z$

Dependency graph:



$C \to Y$

$X \to Z$

$C \to X$ $X \to Y$ $Y \to Z$ $C \to Y$

$C \to Y$ $\to Z$ $\boxed{Y \to a}$

$\to Z$ $\to a$ $C \to Z$

$\boxed{C \to a}$ $\boxed{X \to a}$ $\boxed{C \to a}$

$X \to Z$

$\boxed{X \to a}$

$\therefore$ { $S \to AC$

$A \to a$

$C \to a/b$

$X \to a$

$Y \to a$ $Z \to a$ }

ent # 11 Student ID 30642
19/9/24 Student Name Nikitha

emove useless productions and symbols from the following CFG:

$S \rightarrow AB \mid CA, B \rightarrow BC \mid AB, A \rightarrow a, C \rightarrow aB \mid b$

ion:

1) $S \rightarrow AB$ $S \rightarrow CA$ $B \rightarrow BC$ $C \rightarrow aB$

$\rightarrow aB$ $\rightarrow bA$ ✗ ✗

$\rightarrow aBC$ $\rightarrow ba$

$\rightarrow aBb$

✗

2) $S \rightarrow AB$ $S \rightarrow CA$

✗ $\rightarrow ba$

$\therefore$ $S \rightarrow CA$

$A \rightarrow a$

$C \rightarrow bA$

Experiment # 11 Student ID 30642
Date 10/9/24 Student Name Niflock

3. Remove unit production from the following CFG:

$S \rightarrow 0A \mid 1B \mid C,\ A \rightarrow 0S \mid 00,\ B \rightarrow 1 \mid A,\ C \rightarrow 01$

Solution:

$S \rightarrow C$

$B \rightarrow A$

$S \rightarrow C$ $\quad B \rightarrow A$ $\quad$ (ii) $B \rightarrow 0S \rightarrow D \rightarrow 00A$

$S \rightarrow 01$ $\quad$ (iii) $B \rightarrow 00$ $\quad$ 0000

$\therefore S \rightarrow 0A \mid 1B \mid 0$

$A \rightarrow 0S \mid 00$

$B \rightarrow 1 \mid 00$

$C \rightarrow 01$

11/9/24 3064 NIRUCo

TUTORIAL (To be carried out in presence of faculty in classroom)

Convert the given grammar to CNF

$S \to aAD$, $A \to aB \mid bAB$, $B \to b$, $D \to d$

lution:

1) Simplification of CNF: NIL

2) Already in CNF

$B \to b$

$D \to d$

Introduce two new variables $C_a \to a$

$C_b \to b$

$S \to aAD$ $A \to aB$ $A \to bAB$

$\to C_aAD$ $\to C_aB$ $\to C_bAB$

(CNF)

introduce $(m-2)$ variable D_1,

$S \to C_aD_1$

$D_1 \to AD$

introduce $(m-2)$ variable D_2,

$A \to C_bD_2$

$D_2 \to AB$

$P = \{ S \to C_aD_1$ $C_b \to b$ $G = (\{S, A, B, C_a, C_b, D, D_1, D_2\}, \{a, b\},$

$A \to C_aB \mid C_bD_2$ $D \to d$ $P, S)$

$B \to b$ $D_1 \to AD$

$C_a \to a$ $D_2 \to AB \}$

127

Student ID 30642

Student Name MOHAMED

Date 10/9/24

2.Convert the following CFG to CNF

S →XSY, X→ aXS | a | ε, Y →SbS | X | bb.

Solution:

Step1 :

∈-productions:

X → ∈

S → XSY X → aXS

S → SY X → aS

unit-production :

Y → X

i) → a

Y → a

Useless productions :

1) S → XSY

→ aSbb

(X)

useless : {S}

X → aXS

(X)

Y → SbS

→ (X)

Y → X

→ a

S → SY (CNF)

P = { X → a/aS

Y → Sbb/a }

2) S → XSY

(X)

Let $C_a \to a$, $C_b \to b$

X → aS Y → bb

→ CaS → CbCb

G. ({S, X, Y, Y}, {a, b}, P', S)

$C_a C_b$

P' = { S → SY

X → a/CaS

Y → a/CbCb

$C_a \to a$

$C_b \to b$

Experiment No. 11 Student ID 30642
Date 18/9/24 Student Name Nirshitha

3. Convert the following grammar into CNF.

$S \rightarrow aAB$, $A \rightarrow aB \mid bAB$, $B \rightarrow aBB \mid bS \mid b$.

Solution:

Step1 Simplification: NIL

Step2: $B \rightarrow b$ already CNF

introduce two new variables $C_a \rightarrow a$, $C_b \rightarrow b$

$S \rightarrow C_aAB$

$A \rightarrow C_aB$ (CNF)

$A \rightarrow C_bAB$

$B \rightarrow C_aBB$

$B \rightarrow C_bS$ (CNF)

introduce $(m-2)$ variables D_1, D_2

$S \rightarrow C_aD_1$

$A \rightarrow C_bD_1$

$D_1 \rightarrow AB$

$B \rightarrow C_aD_2$

$D_2 \rightarrow BB$

$G = (\{S, A, B, C_a, C_b, D_1, D_2\}, \{a, b\}, P, S)$

$P = \{ S \rightarrow C_aD_1$

$A \rightarrow C_aB \mid C_bD_1$

$B \rightarrow C_aD_2 \mid C_bS \mid b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D_1 \rightarrow AB$

$D_2 \rightarrow BB \}$

Experiment # 11 Student ID 23003064 42
Date 15/4/24 Student Name Nishal Ca

Post-Tutorial (To be carried out by student after attending tutorial session)

1.Convert the following CFG to CNF: S-->A0B, A-->AA | 0S | 0, B--> 0BB, | 1S | 1

Solution:

$A \to AA$
$B \to 1$ } already CNF
$A \to 0$

let $C_a \to a0$
$C_b \to 1$

$S \to A0B$

$\Rightarrow S \to A C_a B$
$A \to C_a S$ (CNF)
$B \to C_a BB$
$B \to C_b S$ (CNF)

let D_1, D_2 (m-1) variable,
$S \to AD_1$ $B \to C_a D_2$
$D_1 \to C_a B$ $D_2 \to BB$

$G = (\{S, A, B, C_a, C_b, D_1, D_2\}, \{0, 1\}, P, S)$

$P = \{ S \to AD_1$
$A \to AA \mid C_a S \mid 0$
$B \to C_a D_2 \mid C_b S \mid 1$
$C_a \to 0$
$C_b \to 1$
$D_1 \to C_a B$
$D_2 \to BB \}$

2.Convert the following CFG to CNF: S --> AB, A--> aab, B --> aAc

Solution:

$S \to AB$ (already CNF)

let $C_a \to a$, $C_b \to b$, $C_c \to c$
$B \to aAc$
$C_a A C_c$

$A \to C_a C_a C_b$

$G = (\{S, A, B, C_a, C_b, C_c, D_1, D_2\}, \{a, b, c\}, P, S)$

let D_1, D_2 (m-1) variable
$B \to C_a D_1$ $A \to C_a D_2$
$D_1 \to A C_c$ $D_2 \to C_a C_b$

$P = \{ S \to AB$
$A \to C_a D_2$
$B \to C_a D_1$
$C_a \to a$
$C_b \to b$
$C_c \to c$
$D_1 \to AC_c$
$D_2 \to C_a C_b \}$

Experiment # 11 Student ID 3D642
19/9/24 Student Name Nithila

Convert the following CFG into CNF: S → ASA | aB, A → B | S, B → b | ε

Solution: (i) $B \to \varepsilon$

$S \to aB$

$\boxed{S \to a}$

(ii) $A \to B$

$A \to S$

$\boxed{A \to b}$

$A \to aB$

$\boxed{A \to ab}$

(iii) 1) $S \to ASA$ $S \to aB$ $A \to B$ $A \to S$

$\to BaBB$ $\to$ 'ab' $\to$ 'b' $\to AB$

$\to$ 'babb' $\to$ 'ab'

2) $S \to ASA$ $S \to aB$

$P = \{ S \to ASA \mid aB \mid a$

$A \to b \mid ab$

$B \to b \}$

$B \to a$

$A \to b$ } already CNF.

$B \to b$

let $C_a \to a$, $C_b \to b$

$S \to CaB$

$A \to CaCb$

$S \to ASA$

let D_1 (m-2) variable

$S \to AD_1$

$D_1 \to SA$

$P' = \{ S \to AD_1 \mid a \mid CaB$

$A \to CaCb \mid b$

$B \to b$

$Ca \to a$

$Cb \to b$

$D_1 \to SA \}$

$G = (\{S, A, B, Ca, Cb, D\}, \{a, b\}, P', S)$.

Experiment #		Student ID	2300030642
Date	18/9/24	Student Name	Nipuna

Viva Questions

1. What is Chomsky Normal Form (CNF) in the context of context-free grammars, and what are its key characteristics?

Answer:

Any CFG without 'ϵ' is generated by a grammar in which all productions are of the form $A \rightarrow BC$ or $A \rightarrow a$. Here A, B and C are variables and 'a' is terminal.

$V \rightarrow VV$

$V \rightarrow T$

2. Why is it necessary to remove ε-productions when converting to Chomsky Normal Form?

Answer:

Let G be any CFG with ϵ is not in L(G) then there exist an equivalent grammar 'G' having no ϵ-productions.

(For Evaluator's use only)

	Comment of the Evaluator (if Any)	Evaluator's Observation
		Marks Secured: out of 50
		Full Name of the Evaluator:
		Signature of the Evaluator Date of Evaluation:

Experiment #	12	Student ID	2060r
Date	2019	Student Name	S. K. S Reddy

TUTORIAL SESSION 12:

Normal Form: GNF

Concept Building

Greibach Normal Form (GNF)

It is a specific type of context-free grammar that simplifies the structure of production rules to facilitate parsing and theoretical analysis. It is named after Sheila Greibach, an American computer scientist who made significant contributions to formal language theory.

Characteristics of Greibach Normal Form

A context-free grammar is in Greibach Normal Form if every production rule adheres to the following form:

$A \rightarrow a\alpha$

Here, A is a non-terminal symbol, 'a' is a terminal symbol, and α (alpha) is a (possibly empty) string of non-terminal symbols. This form ensures that each production rule begins with a terminal symbol followed by zero or more non-terminals.

Importance of Greibach Normal Form

Algorithmic Parsing: GNF is particularly useful for constructing top-down parsers, such as recursive descent parsers, because it guarantees that each production begins with a terminal symbol, making it easier to decide which rule to apply based on the next input symbol.

Theoretical Analysis: GNF simplifies the analysis of context-free grammars by providing a uniform structure for production rules. This uniformity aids in proving properties about languages and grammars, such as closure properties and decidability.

Equivalence to Other Forms: Any context-free grammar can be converted to an equivalent GNF grammar that generates the same language. This conversion preserves the language while restructuring the grammar for easier parsing.

Steps to Convert a Context-Free Grammar to GNF

Converting a context-free grammar to GNF involves several steps, often requiring intermediate transformations and the introduction of new non-terminals.

Remove Left Recursion: Ensure that the grammar is free from left recursion, as left-recursive rules cannot be converted directly to GNF.

Eliminate Non-Productive and Unreachable Symbols: Remove any non-productive symbols (symbols that do not derive any terminal strings) and unreachable symbols (symbols that cannot be reached from the start symbol).

Ensure Proper Ordering: Ensure that the non-terminals in the grammar are ordered such that when converting each rule to GNF, the right-hand side contains only non-terminals that have already been processed.

Experiment #	12	Student ID	30604
Date	20/9	Student Name	S.R.S. Reddy

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Convert the given CFG to GNF:

S→AB, A→BS | b, B→ SA | a

V → T

V → TV'

Solution:

(i) No epsilon transitions

(ii) No units transitions

(iii) S → AB A → b B → a

→ ba

{S, A, B}

A → BS

SAS

S → AB

→ ba {S, A, B}

(iv) A → b, B → a are GNF

S → AB	S → bB ✓
A → BS	A → aS
B → SA	B → Sb

(v) GNF

S → AB

A → aAB | b

B → bBA | a.

pl: { A → b | aS

B → Sb | b.

S → bB }.

Experiment #	12	Student ID	24604
Date	20/9	Student Name	S.K.Sreddy

2.Transform the following grammar into GNF. S→AA | 0, A→SS | 1

Solution:

Simplification

1) Epsilon transitions

P: {S→AA|0

A→SS|1}

No Epsilon

2) No unit transition

3) Useless transitions

S→AA A→SS

S→'11' →'00'

4) GNF

S→AA

⇒ S→SSSS|SS1|1SS|11

A→SS

⇒ A→AAAA|AA0|0AA|00.

Final:

P:{ S→0|1SS|SS1|11|SSSS

A→00|0AA|AA0|AAAA }.

Experiment #	12.	Student ID	30627
Date	20/9	Student Name	G.R.S. Reddy

3.Find GNF equivalent for the given CFG:

S→ CA | BB, B→ b | SB, C→b, A→a

Solution:

Simplification

$P:\{S \rightarrow CA/BB$

$B \rightarrow b/SB$

$C \rightarrow b$

$A \rightarrow a\}.$

1) Epsilon transitions

No epsilon

2) Unit transition

No Unit transitions

3) Usesless transitions

$S \rightarrow CA \quad B \rightarrow 'b' \quad C \rightarrow 'b' \quad A \rightarrow 'a'$

$\rightarrow 'ba'$

4) GNF:

$P': \{ S \rightarrow ba/bb$

$B \rightarrow b/baB/bbB$

$C \rightarrow b$

$A \rightarrow a\}.$

Experiment #	12	Student ID	3062..7
Date	20/7	Student Name	G. P. S. Varthy

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Convert the following grammar into Greibach Normal Form:

$E \to E+T \mid T,\ T \to T^*F \mid F,\ F \to (E) \mid a$

Solution:

Given

$S: \{ E \to E+T \mid T$

$T \to T^*F \mid F$

$F \to (E) \mid a \}.$

Simplification

(i) No Epsilon

(ii) Unit

$E \to T$ $E \to F$

$T \to F$ ✓

$E \to T^*F$ ✓

$T \to a \mid (E)$ ✓

$E \to a \mid (E)$ ✓

$\Rightarrow E \to E+T \mid a \mid (E) \mid T^*F$

$T \to T^*F \mid a \mid (E)$

$F \to (E) \mid a \}.$

(iii) Remove useless transition.

$E \to a \quad T \to a \quad F \to a$

No useless

(iv) Final GNF

$S: \{ E \to FT'E'$

$E' \to +FT'E' \mid$

$T \to FT'$

$T' \to *FT'$

$F \to (FT'E') \mid a \}.$

Experiment #	12	Student ID	30691
Date	20/9	Student Name	S. B. S. Reddy

2.Convert the following CFG to GNF S → AB, A→ BS | a, B → SA | b.

Solution:

P: {S → AB

A → BS | a

B → SA | b }.

Simplification of CFG

(i) No nullable values

∴ No Epsilon transitions

(ii) Unit transitions

∴ No Unit transitions

(iii) Remove Useless transitions

S → AB A → a B → b

→ ab

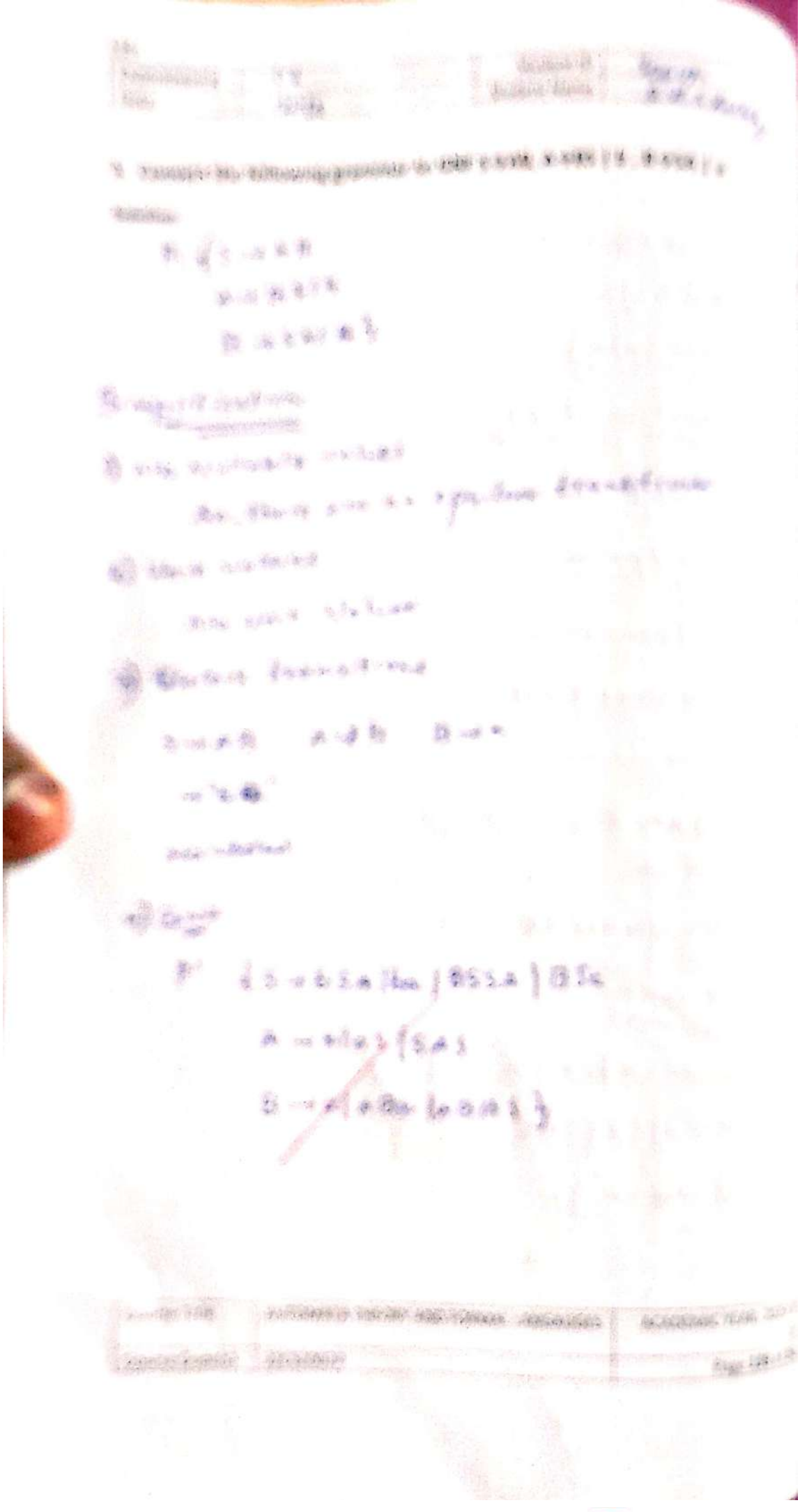
∴ No useless transitions

(iv) Final GNF

P: {S → aSA | ab | BSSA | BSb

A → a | SAS | bS

B → b | SA }.



Experiment #	12	Student ID	2060
Date	20/9	Student Name	S.P.S.P

4. A grammar G is defined with rules $S \rightarrow XA \mid BB, B \rightarrow b \mid SB, X \rightarrow b, A \rightarrow a$. Write the productions obtained after normalized GNF of G.

Solution:

$V \rightarrow T$

$V \rightarrow TV'$

P: { $S \rightarrow XA \mid BB$

$B \rightarrow b \mid SB$

$X \rightarrow b$

$A \rightarrow a$ }.

Simplification

1) Epsilon transitions

No Nullable Values

2) Unit Values

No Unit Values

3) Useless transitions

$S \rightarrow XA \quad B \rightarrow b \quad X \rightarrow b \quad A \rightarrow a$

$\rightarrow$ 'ba'

4) GNF

Already GNF

$X \rightarrow b$ ✓

$A \rightarrow a$ ✓

$B \rightarrow b$ ✓

Final GNF

P1 : { ~~$S \rightarrow bSA \mid ba \mid BSSA \mid bS \mid a$~~

$S \rightarrow ba \mid bB \mid SBB$.

$B \rightarrow b \mid baB \mid bBB \mid aSBBB$

$X \rightarrow b$

$A \rightarrow a$ }.

Experiment #	12	Student ID	36604
Date	2/9	Student Name	S.12.S.Reddy

Post-Tutorial (To be carried out by student after attending tutorial session)

1. **Convert the following CFG to GNF: S→ AB, A→ BSB, A→ a, B→ b**

Solution:

P: {S → AB

A → BS | B

A → a

B → b}

Simplification

1) Epsilon transition

No nullable values.

2) Unit productions

No Unit productions

3) Useless

S → AB A → BSB B → b.

→ 'ab' → bABb

→ babb

4) GNF

Already GNF Find GNF

A → a

B → b

P': {S → ba | bB | SBB

B → b | baB | bBB

X → a

A → a }

Experiment #	12.	Student ID	30601
Date	20/7	Student Name	S.R.S. Reddy

2. Convert the following CFG to GNF : S→ A, A→ aBa | a, B → bAb | b

Solution:

P: {S → A
A → aBa | a
B → bAb | b }

Simplification

1) No epsilon

2) No Unit

3) Useless

S → A A → a
→ a' B → b

1) GNF

Already GNF

A → a
B → b

Final

P1 : {S → aBa | a
A → aBa | a
B → baBab | bab | b }.

3. Write the steps for removing null productions and unreachable symbols? Explain with an example of your own.

Solution:

1. Removing null productions

- Identify nullable non-terminals.
- For each nullable non-terminal, add new productions
- Remove the null productions.

2) Remove Unreachable Symbols

- Identify reachable symbols by starting from the start symbol and marking symbols used in productions

Experiment #	12	Student ID	30607
Date	20/9	Student Name	S.V.S.Reddy

Viva Questions

1. What is Greibach Normal Form (GNF) in the context of context-free grammars, and what are its characteristics?

Answer:

GNF is a special form of context-free grammars where every production rule starts with a terminal followed by zero or more terminals.

2. What are some practical applications of Greibach Normal Form in computer science?

Answer:

1. Top-Down parsing
2. Theoritical Analysis
3. Proving Grammer Correctness

Experiment #	12	Student ID	30609
Date	20/9	Student Name	S.R.S.Reddy.

3. Can you describe the process of converting a context-free grammar into Greibach Normal Form?

Answer:

To convert a CNF to GNF

1) Start by removing left recursion

2) ensure every production starts with a terminal

$V \rightarrow T$

$V \rightarrow TV^*$

3) Replacing non-terminals with proper terminal first rules

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: 30 out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: 21/9/24

Pre-Tutorial (To be completed by student before attending tutorial session)

1. **Consider the following PDA**

   δ(q0, a, z0) = (q0, Xz0)
   δ(q0, a, X) = (q0, XX)
   δ(q0, b, X) = (q1, ε)
   δ(q1, b, X) = (q1, ε)
   δ(q1, ε, z0) = (q1, ε)

   Convert this PDA to CFG.

Solution:

Define each variable Aq_iq_j, representing strings that have the PDA from state q_i with an empty stack to q_j

$Aq_0q_0 \longrightarrow aAq_0q_0X$

$Aq_0q_0 \longrightarrow \epsilon$

$Aq_0q_1 \longrightarrow aAq_0q_0b$

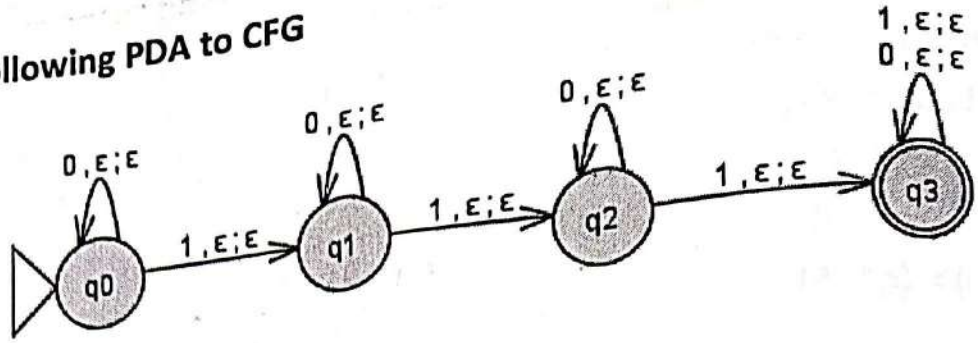
$Aq_1q_1 \longrightarrow bAq_1q_1$

$Aq_1q_1 \longrightarrow \epsilon$

CFG for given PDA is

$S \longrightarrow aSb \mid \epsilon$

3. Convert the following PDA to CFG



Solution:

① $\delta(q_0, 0, \epsilon) = (q_0, \epsilon)$

$\delta(q_0, 1, \epsilon) = (q_1, \epsilon)$

$\delta(q_0, 0, \epsilon) = (q_1, \epsilon)$

$\delta(q_1, 1, \epsilon) = (q_2, \epsilon)$

$\delta(q_2, 0, \epsilon) = (q_2, \epsilon)$

$\delta(q_2, 1, \epsilon) = (q_3, \epsilon)$

$\delta(q_3, 0, \epsilon) = (q_3, \epsilon)$

$\delta(q_3, 1, \epsilon) = (q_3, \epsilon)$

② Aq_0q_1 = q_0 to q_1

Aq_1q_2 = q_1 to q_2

Aq_2q_3 = q_2 to q_3

$Aq_0q_0 \neq Aq_1q_1$: Self loop

Aq_0q_0, Aq_1q_1 : self loops

③ Create CFG Rules:

$Aq_0q_1 \rightarrow 1$

$Aq_1q_2 \rightarrow 1$

$Aq_2q_3 \rightarrow 1$

$Aq_0q_0 \rightarrow 0Aq_0q_0 / \epsilon$

$Aq_1q_1 \rightarrow 0Aq_1q_1 / \epsilon$

$Aq_2q_2 \rightarrow 0Aq_2q_2 / \epsilon$

④ $S = Aq_0q_3$

⑤ $S \rightarrow Aq_0q_1, Aq_1q_2, Aq_2q_3$

$Aq_0q_0 \rightarrow 0Aq_0q_0 / \epsilon$

$Aq_1q_1 \rightarrow 0Aq_1q_1 / \epsilon$

$Aq_2q_2 \rightarrow 0Aq_2q_2 / \epsilon$

$Aq_3q_3 \rightarrow 0Aq_3q_3 / 1Aq_3q_3 / \epsilon$

$Aq_3q_3 \rightarrow 1$

$\Leftrightarrow Aq_0q_1 \rightarrow 1$

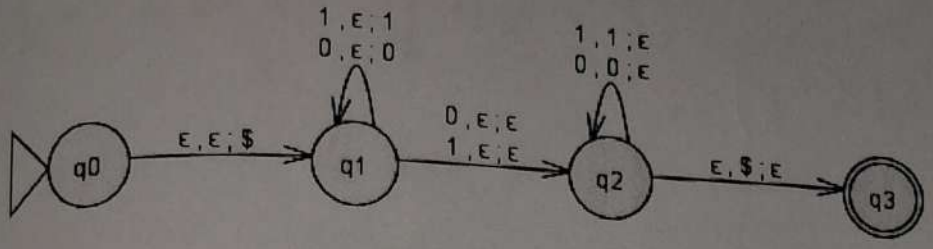
$- Aq_1q_2 \rightarrow 1$

$- Aq_2q_3 \rightarrow 1$

Final CFG: $S \rightarrow 111 / 0S1 / 0 / \epsilon$

(Non-TUTORIAL (To be carried out in presence of faculty in classroom)

1. **Convert the following PDA to CFG.**



Solution:

$(q_1, 1, \epsilon) \rightarrow (q_1, 1)$

$(q_1, 0, \epsilon) \rightarrow (q_1, 0)$

$(q_1, 0, 0) = (q_2 \rightarrow \epsilon)$

$(q_1, 1, 1) = (q_2 \rightarrow \epsilon)$

$(q_2, 0, 1) \rightarrow (q_2 \rightarrow \epsilon)$

$(q_2, \epsilon, \$) \rightarrow (q_3, \epsilon)$

$S \rightarrow A q_3$

Date

2. Let L be the language $\{0^n 1^{2n} \mid n \geq 1\}$. For the given language ... the PDA.

Solution:

$L = \{0^n 1^{2n} / n \geq 1\}$

$= \{011, 001111, 000111111, 00001111111, \ldots\}$

1) $\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$ } 0's

2) $\delta(q_0, 0, 0) = (q_0, 00)$

3) $\delta(q_0, 1, 0) = (q_1, 0)$ } rejected

4) $\delta(q_1, 1, 0) = (q_2, \epsilon)$ }

5) $\delta(q_2, 1, 0) = (q_1, 0)$

6. $\delta(q_1, \epsilon, Z_0) = (q_3, Z_0)$

$(q_1, \epsilon, Z_0) = (q_3, Z_0)$ ⟶ accepted

$\therefore q_3 \in F$

0 — push '0'

1 = { q_1 - '1' - no. op - change q_2' }

$M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \{0, Z_0\}, \delta, q_0, Z_0, \{q_3\})$

3.Construct a PDA for language L = {0^n1^m | n >= 1, m >= 1, m > n+2}

Solution:

States :—

q_0 → Initial states

q_1 → (read 0s and push)

q_2 → (read 1s and pop)

q_3 (ensure extra 1s)

q_4 → final state

Transitions :

Reading 0's :

$\delta(q_0, 0, \epsilon) = (q_1, x)$ — push x for each x & move to q_1 } 0's

$\delta(q_1, 0, \epsilon) \rightarrow (q_1, x)$

$\delta(q_1, 1, x) = (q_2, \epsilon)$ — pop 1 x for each 1

$\delta(q_2, 1, x) = (q_2, \epsilon)$ — pop 1 x for each ① } is repeat

$\delta(q_2, 1, x) = (q_2, \epsilon)$ — pop 1 x for each 1 }

$\delta(q_2, 1, x) = (q_2, \epsilon)$ — pop x for each 1

$\delta(q_2, 1, \epsilon) \rightarrow$ if stack is empty move to q_3 & read additional acceptance.

$\delta(q_3, 1, \epsilon) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, \epsilon) = (q_4, \epsilon)$

Initial state → q_0

final state → q_4

4. Design a PDA for accepting a language $\{a^n b^m c^{(m+n)} \mid m, n >= 1\}$.

Solution:

{ abcc , aaabbbb ccccccc , ---- }

push (aaa) pop (bbbb) ccc ccc c

$\delta(q_0, a, \epsilon) = (q_1, A)$
$\delta(q_1, a, \epsilon) = (q_1, A)$ } a's
$\delta(q_1, b, \epsilon) = (q_2, B)$

$\delta(q_2, b, \epsilon) = (q_2, B)$
$\delta(q_2, c, A) = (q_3, \epsilon)$ } b's
$\delta(q_2, c, B) = (q_3, \epsilon)$

$\delta(q_3, c, A) = (q_3, \epsilon)$
$\delta(q_3, \epsilon, B) = (q_3, \epsilon)$ } c's
$\delta(q_3, \epsilon, \epsilon) = (q_4, \epsilon) \Rightarrow$ acceptance

Initial state : q_0
final state : q_4

Date

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Construct a PDA that accept the language L = {0^n1^n | n > 0 }

Solution:

$\delta(q_0, 0, \epsilon) = (q_0, x)$ — 0's

$\delta(q_0, 1, x) = (q_1, \epsilon)$

$\delta(q_1, 1, x) = (q_1, \epsilon)$

$\delta(q_1, 1, x) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, \epsilon) = (q_2, \epsilon)$ - acceptance

Initial state = q_0

final state = q_2

$L = \{01, 0011, \ldots\}$

~~0~~ ~~0~~ 0011 ~~1~~ ~~1~~

for every 0 pop 1

2. Construct a PDA from the following CFG. G = ({S, X}, {a, b}, P, S) where the productions are –

S → XS | ε ,

A → aXb | Ab | ab

Solution:

$G = (\{S, X\}, \{a, b\}, P, S)$

$S \rightarrow XS \mid \epsilon$

$A \rightarrow aXb \mid Ab \mid ab$

$\delta(q_0, \epsilon, \epsilon) = (q_1, S)$ — push

$S \rightarrow XS$:

$\delta(q_1, \epsilon, \delta) = (q, XS)$

$S \rightarrow \epsilon$: $\delta(q_1, \epsilon, d) = (q_1, \epsilon)$

$X \rightarrow aXb$: $\delta(q, \epsilon, X) = (q_1, \epsilon)$

$X \rightarrow Ab$: $\delta(q_1, \epsilon, X) = (q_1, Ab)$

$X \rightarrow ab$: $\delta(q, \epsilon, X) = (q, ab)$

→ $\delta(q_1, a, a) = (q_1, \epsilon)$

– $\delta(q_1, b, b) = (q_1, \epsilon)$

– $\delta(q_1, \epsilon, \epsilon) = (q_2, \epsilon)$ ⇒ Acceptance

q_0 → Initial state

q_2 → final state.

3.Construct a PDA for language L = {$0^n1^m2^m3^n$ | n>=1, m>=1}

Solution:

L = {0123, 00112233, ...}.

$\delta(q_0, 0, \epsilon) = (q_0, x)$

$\delta(q_0, 1, \epsilon) = (q_1, \epsilon)$ } 0's

$\delta(q_1, 1, \epsilon) = (q_1, y)$

$\delta(q_1, 2, Y) = (q_2, \epsilon$ } 1's

$\delta(q_2, 2, 4) = (q_2, \epsilon)$

$\delta(q_2, 3, x) = (q_3, \epsilon)$ } 2's

$\delta(q_3, 3, x) = (q_3, \epsilon)$ — 3's

$\delta(q_3, \epsilon, \epsilon) = (q_4, \epsilon)$ — acceptance.

Initial state — q_0

final state — q_4

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva Question:

1.Write the instantaneous Description (ID) of PDA?

Solution:

The instantaneous Description (ID) of a PDA provides a snap shot its current Configuration.

2.What are the components of PDA?

Solution:

I/p tape: A sequence of symbols from a finite alphabet

Stack: A storage structure that allows the PDA to store an Arbitrary amount of information

4.Write the applications where the PDA used in real time?

Solution:

Syntax parsing: used in compilers & Interpreters for programming language to parse expressions & syntax trees.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Course Title	AUTOMATA THEORY AND FORMAL LANGAUGES	ACADEMIC YEAR 2023-24

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Convert the following CFG to a PDA:

$S \rightarrow aSb \mid \varepsilon$

Solution:

$Z \rightarrow$ Stack symbol

$A \rightarrow a$

$q_0 \rightarrow$ initial state

$(q_0, a, Z) \rightarrow (q_0, AZ)$

$(q_0, a, A) \rightarrow (q_0, AA)$

$(q_0, b, A) \rightarrow (q_0, \epsilon)$

$(q_0, \epsilon, Z) \rightarrow (q_f, Z) \Rightarrow$ acceptance

$q_f \rightarrow$ final state

PDA accepts empty ~~state~~ stack, mean has no remaining unmatched 'a's in the stack.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Prove that the context-free language $L = \{a^n b^n \mid n \geq 0\}$ is closed under concatenation.

Solution:

$L = \{ \in, ab, aabb, \ldots \}$

$x = a^{n_1} b^{n_1}$ $\quad y = a^{n_2} b^{n_2} \rightarrow n_1, n_2 \geq 0$

$x \in L$ $\quad y \in L$

Concatenation of $xy = a^{n_1} b^{n_1}, a^{n_2} b^{n_2}$

$xy = a^{n_1} b^{n_1}, b^{n_1} b^{n_2}$ $\rightarrow a's = n_1 + n_2$, $b's = n_1 + n_2$

$a^{(n_1+n_2)} \; b^{(n_1+n_2)} \rightarrow a^m b^m \; (m = n_1 + n_2)$

$m \geq 0$, $xy = a^m b^m$ for some m,

hence $xy \in L$

3. Use the Pumping Lemma to prove that the language $L = \{a^n b^n c^n \mid n \geq 0\}$ is not a context free language.

Solution:

$L = \{\epsilon, abc, aabbcc, aaabbbccc, \ldots\}$

S1: Assume L is CFL $\boxed{n=4}$ $k=2$

S2: Choose a string z from L, $|z| \geq n$

$6 \geq 4$

S3: $z = uvwxy$, $|vx| \geq 1$

$|vwx| \leq n$

$z = aabbcc$ $|vx| \geq 1$ $|vwx| \leq n$

$2 \geq 1$ $4 \leq 4$

S4: Find a integer $i = 0, 1, 2, 3, 4 \ldots$,

$uv^i wx^i y \notin L$

$i = 2$ $a(a)^2 bb (c)^2 c$

$= a^3 b^2 c^3 \notin L$

Our Assumption is wrong.

∴ The given language is not CFL

TUTORIAL (To be carried out in presence of faculty in classroom)

Given the CFG:

$$S \rightarrow aA \mid bB$$
$$A \rightarrow aS \mid bAA \mid \varepsilon$$
$$B \rightarrow bS \mid aBB \mid \varepsilon$$

Construct the equivalent PDA.

ution: $\Sigma = \{a, b\}$ $\Gamma = \{S, A, B, Z\}$

$(q_0, \epsilon, Z) \rightarrow (q_0, SZ)$

S: $(q_0, \epsilon, S) \rightarrow (q_0, A)$

$(q_0, \epsilon, S) = (q_0, B)$

A: $(q_0, \epsilon, A) \rightarrow (q_0, aS)$

$(q_0, \epsilon, A) \rightarrow (q_0, bAA)$

$(q_0, \epsilon, A) \rightarrow (q_0, \epsilon)$

S: $(q_0, \epsilon, B) \rightarrow (q_0, bS)$

$(q_0, \epsilon, B) \rightarrow (q_0, aBB)$

$(q_0, \epsilon, B) \rightarrow (q_0, \epsilon)$

$(q_0, a, a) \rightarrow (q_0, \epsilon)$

$(q_0, b, b) \rightarrow (q_0, \epsilon)$

PDA recognizes the language generated by given CFG.

2. Show that the language $L = \{a^n b^m \mid n \neq m\}$ is context-free.

Solution:

$L = \{a^n b^m \mid n \neq m\}$

$S \rightarrow A/B$

$A \rightarrow aA/aB/bB/b \quad (n>m)$

$B \rightarrow bB/aB/aA/a \quad (m>n)$

$S \rightarrow A$ (String with more a's than b's)

$S \rightarrow B$ (String with more b's than a's)

$a's \neq b's$

So, L is content free.

3. Consider the CFL L1 = {$a^n b^n$ | n ≥ 0} and L2 = {$b^n a^n$ | n ≥ 0}. Prove that L1 ∪ L2 is also a context-free language.

Solution:

$L_1 = \{a^n b^n / n \geq 0\}$ $L_2 \{b^n a^n / n \geq 0\}$

$L_1 \cup L_2$

$S \rightarrow A/B$

$A \rightarrow aAb/\epsilon$ (generates strings $a^n b^n$)

$B \rightarrow bBa/\epsilon$ (generates strings $b^n a^n$)

CFG generates all strings in L_1 and L_2 thus proving $L_1 \cup L_2$ is a context-free language.

$L_1 = \{a^n b^n / n \geq 0\}$

$G_1 = (\{S_1\}, \{a, b\}, P_1, S_1)$

$P_1 = \{S_1 \rightarrow aS_1 b / ab\}$

$\therefore L_1$ is CFL

$L_2 = \{\omega\omega^R / \omega \in \{0,1\}^*\}$

$G_2 = (\{S_2\}, \{0,1\}, P_2, S_2)$

$P_2 = \{S_2 \rightarrow 0S_2 0 / \{S_2\} / \epsilon\}$

$\therefore L_2$ is CFL

$G = (\{S, S_1, S_2\}, \{a, b, 0, 1\}, P, S)$

$P = \{S \rightarrow S_1 / S_2$

$S_1 \rightarrow aS_1 b / ab$

$S_2 \rightarrow 0S_2 0 / 1S_2 1 / \epsilon\}$

$\therefore L$ is CFL

4. If L is a CFL and R is a regular language, then what can be L ∩ R? Will it be CFL or R... Why?

Solution:

$L_1 = \{a^i b^i c^j / i, j \geq 1\}$ $\quad L_2 = \{a^i b^j c^j / i, j \geq 1\}$

$P_1 = \{S_1 \rightarrow AB$

$A \rightarrow aAb / ab$

$B \rightarrow cB / c\}$

$P = \{S_2 \rightarrow MN$

$M \rightarrow aM / a$

$N \rightarrow bNc / bc\}$

$L = L_1 \cap L_2 \rightarrow$ CFL ??

$= \{a^n b^n c^n / n \geq 1\}$

$\therefore L$ is not CFL

d on your answer for Q#4 consider the languages:

$$L1 = \{0^n1^n2^i : n \geq 1, i \geq 1\}$$

$$L2 = \{0^i1^n2^n : n \geq 1, i \geq 1\}$$

se context-free languages? If yes, will the language L = L1 ∩ L2 be context-free? e language L.

$L_1 = \{0^n 1^n 2^i \mid n \geq 1, i \geq 1\}$

$L_2 = \{0^i 1^n 2^n \mid n \geq 1, i \geq 1\}$

$S \rightarrow 0S \mid 1A$

$A \rightarrow 1A2 \mid 12$

$0^n 1^n 2^i \; (n \geq 1, i \geq 1)$

$S \rightarrow 0S \mid 1A$

$A \rightarrow 1A2 \mid 12$

$0^n 1^n 2^n \; (n \geq 1, i \geq 1)$

$L = \{0^n, 1^n, 2^n \mid n \geq 1\}$

$\therefore$ L' is not CFL

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Convert the following CFG into an equivalent PDA:

$S \rightarrow aS \mid bA$

$A \rightarrow bS \mid \varepsilon$

Solution:

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

$\Gamma = \{S, A, Z\}$

$(q_0, \epsilon, Z) \rightarrow (q_0, SZ)$

$(q_0, \epsilon, S) \rightarrow (q_0, aS)$

$(q_0, \epsilon, S) \rightarrow (q_0, bA)$

$(q_0, \epsilon, A) \rightarrow (q_0, bS)$

$(q_0, \epsilon, A) \rightarrow (q_0, bS)$

$(q_0, \epsilon, A) \rightarrow (q_0, \epsilon)$

$\rightarrow (q_0, a, a) \rightarrow (q_0, \epsilon)$

$(q_0, b, b) \rightarrow (q_0, \epsilon)$

PDA recognizes the language generated by the given CFG.

ermine if the language $L = \{ww^R \mid w \in \{a, b\}^*\}$ is a context-free language and justify nswer.

$L = \{ww^R / w \in \{a,b\}^*\}$

$G = \{\{S\}, \{a,b\}, P, S\}$

$P = \{S \rightarrow 0S0 / 1S1 / \in\}$

$\therefore L$ is CFL

3. Use the Pumping Lemma to prove that the language $L = \{a^n b^n c^m \mid n \neq m\}$ is not a context free language.

Solution:

$S = a^P b^P c^{P+1}$ $(n = P, m = P+1)$

$S = uvwxy$ where $|vwx| \leq P$ and

$|vx| \geq 1$

vwx include a's / only b's.

$n \neq m$

$\rightarrow$ v has b's & x has b's

$n = m$, violating $n \neq m$

we cannot maintain $n \neq m$

'L' is not content free

Student Name	<TO BE FILLED BY STUDENT>
	<TO BE FILLED BY STUDENT>

. questions

Can a language be generated by a context-free grammar but not recognized by a pushdown automaton? y your answer.

on:

No, a language cannot be generated by a CFG but not recognized by a pushdown automata (PDA). Thus, if a language is generated by a CFG it is guaranteed to be recognized by a PDA

What is the role of the stack in a pushdown automaton, and how does it relate to context-free grammars?

lution:

The stack in a PDA serves as memory for tracking symbols, enabling the PDA to handle nested structures.

r Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Assignment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date		Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. **Define computability.**

Solution: Computability refers to the study of what problems can be solved by a computational process and the limits of what can be computed. More formally its a branch of theoritical computer science and mathematical logic that explores the capabilities & limitations

2. **In what real-world scenarios do you think Turing machines or their principles are applied today? Can you identify any algorithms or technologies that are based on Turing machine concepts?**

Solution:

Programming languages - universal computation guide how compilers and interpreters work.

AI and ML Algorithms process data and make decisions, inspired by logical steps of turing machines.

3. **Design a Turing machine that accepts strings if and only if they are palindrome the alphabet {0, 1}.**

Solution:

$L = \{010, 101, 001100, 1001, \ldots\}$

i/p :- BB010BB

	δ	0	1	X	Y	B
→	q_0	(q_1, X, R)	(q_2, Y, R)	(q_0, X, R)		
	q_1	(q_3, X, L)	$(q_1, 1, R)$			
	q_2				(q_4, X, L)	
	q_3		$(q_3, 1, L)$	(q_0, X, R)		
	q_4					$(q_5, 4, L)$
*	q_5					

initial state → q_0 final state q_5

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_5\})$

TUTORIAL (To be carried out in presence of faculty in classroom)

Construct a Turing Machine for the language $L = \{a^n b^n c^n \mid n \geq 0\}$.

Soln: $\{\epsilon, abc, aabbcc, aaabbbccc \ldots\}$

tape Symbols

S	a	b	c	X	Y	Z	B
q_0	(q_1, X, R)				(q_4, Y, R)		(q_5, B, R)
q_1	(q_1, a, R)	(q_2, Y, R)			(q_1, Y, R)		
q_2		(q_2, b, R)	(q_3, Z, Y)			(q_2, Z, R)	
q_3	(q_3, a, L)	(q_3, b, L)			(q_3, Y, L)	(q_3, Z, L)	
q_4					$(q_4, 4, R)$	(q_4, Z, R)	(q_5, t, R)
q_5	—	—	—		—	—	—

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{a, b, c, X, Y, Z, B\}, \delta, q_0, B, \{q_5\})$

2. **Design a Turing machine that takes a binary number as input (e.g., 101) increments it by 1 (e.g., outputs 110). Assume the input is given with the significant bit on the left.**

Solution:

δ	0	1	B
$\rightarrow q_0$	$(q_1, 1, R)$	$(q_0, 0, R)$	$(q_1, 1, R)$
$*\ q_1$	—	—	—

Initial state - q_0, final state - q_1

$M = (\{q_0, q_1\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_1\})$

...epts binary numbers divisible by 3.

δ	0	1	X	Y	B
q_0	(q_0, X, R)	$(q_1, 4, R)$	(q_0, X, R)	(q_1, Y, R)	$(q_3 B, R$
q_1	(q_2, X, R)	(q_0, Y, R)	(q_2, X, R)	$(q_B, 4, R)$	(q_3, B, R)
q_2	(q_1, X, R)	(q_2, Y, R)	(q_3, B, R)	(q_2, Y, R)	(q_3, B, R)
* q_3	—	—	—	—	—

$M = (\{q_0, q_1, q_2, q_3\} : \{0, 1\}, \{0, 1, X, Y, \$\}, \delta q_0,$
$B, \{q_3\})$.

4. Construct a Turing Machine that ... input ...
011, the output should be 011011.

Solution:

δ	0	1	X	Y	B
→ q_0	(q_1, X, R)	(q_2, Y, R)	(Error)	(Error)	(q_1, B, R)
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	E	E	$(q_4, 0, L)$
q_2	$(q_2, 0, R)$	$(q_2, 1, R)$			
q_3	E	E	$(q_3, 0, R)$	$(q_3, 1, R)$	(q_6, B, L)
q_4	$(q_4, 0, L)$	$(q_4, 1, L)$	(q_0, X, R)	E	E
q_5	$(q_5, 0, L)$	$(q_5, 1, L)$	E	(q_0, Y, R)	E
q_6	E	E	$(q_6, 0, L)$	$(q_6, 1, L)$	(q_7, B, R)
* q_7	—	—	—	—	—

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{0, 1\},$
$\{0, 1, X, Y, B\}, q_0, \delta, B, \{q_7\})$

4. Construct a Turing Machine that copies a binary string. For example, if the input 011, the output should be 011011.

Solution:

δ	0	1	X	Y	B
$\rightarrow q_0$	(q_1, X, R)	(q_2, Y, R)	(Error)	(Error)	(q_1, B, R)
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	E	E	$(q_4, 0, L)$
q_2	$(q_2, 0, R)$	$(q_2, 1, R)$			
q_3	E	E	$(q_3, 0, R)$	$(q_3, 1, R)$	(q_6, B, L)
q_4	$(q_4, 0, L)$	$(q_4, 1, L)$	(q_0, X, R)	E	E
q_5	$(q_5, 0, L)$	$(q_5, 1, L)$	E	(q_0, Y, R)	E
q_6	E	E	$(q_6, 0, L)$	$(q_6, 1, L)$	(q_7, B, R)
$*q_7$	—	—	—	—	—

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{0, 1\}, \{0, 1, X, Y, B\}, q_0, \delta, B, \{q_7\})$

utorial (... out by student after attending tutorial session)

Create a Turing Machine to decide the language L = {0^m 1^n | m ≠ n}

:

$	0	1	X	Y	B
q_0	(q_1, X, R)	(q_2, Y, R)	E	E	(q_5, B, R)
q_1	$(q_1, 0, R)$	(q_3, Y, R)	E	(q_1, Y, R)	(q_4, B, L)
q_2	(q_3, X, R)	$(q_2, 1, R)$	E	E	(q_4, B, L)
q_3	$(q_4, 0, L)$	$(q_4, 1, L)$	(q_4, X, L)	(q_4, Y, L)	(q_0, B, R)
q_4	—	—	—	—	—
q_5					

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0, 1\}, \{0, 1, X, Y, B\}$
$\delta, q_0, B, \{q_5\})$

2. **Construct a Turing Machine that replaces every 0 in a binary string with 1 and every with 0 (binary negation).**

Solution:

δ	0	1	X	Y	B
$\rightarrow q_0$	(q_1, X, R)	(q_2, Y, R)	$\in$	$\in$	(q_5, B, R)
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	$\in$	$\in$	$(q_3, 1, R)$
q_2	$(q_2, 0, R)$	$(q_2, 1, R)$	$\in$	$\in$	$(q_4, 0, R)$
q_3	$\in$	$\in$	$(q_0, 1, L)$	$(q_0, 1, L)$	(q_5, B, R)
q_4	$\in$	$\in$	$(q_0, 0, L)$	$(q_0, 0, L)$	(q_5, B, R)
$*q_5$	—	—	—	—	—

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_5\})$

sign a Turing Machine that accepts the language L = {a^n b^m c^k | n + m = k}.

	a	b	X	Y	B
	(q_1, X, R)			(q_0, Y, R)	(q_5, B, R)
	(q, a, R)	(q_2, Y, R)		(q_1, Y, R)	(q_5, B, L)
		(q_2, b, R)	(q_3, Y, R)	(q_2, Y, R)	$(q_5 B, L)$
			(q_3, C, R)	$(q_4, 4, L)$	(q_4, B, L)
q_4	(q_4, a, L)	(q_4, b, L)		(q_0, X, R)	$(q_4, 4, L)$
q_5	—				

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{a, b, c, x, y, B\}, \delta, q_0, B, \{q_5\})$$

Date

Viva – Questions:

1. Describe the components of a Turing machine.

Solution:

A Turning machine consists of an Infinite tape & read-write head (to read & write symbols) a finite set of states, transition function.

2. What is the difference between a deterministic Turing machine (DTM) and a non-deterministic Turing machine (NDTM)?

Solution:

A deterministic turning machine has a single possible action for each state and input, following a predictable path. a Non-deterministic Turing Machine can choose from multiple actions for a state and input.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	
Date	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Design Turing machine for 1's complement of the binary numbers.

Solution:

i/p - 0110 → o/p :- 1001

	0	1	B
→ q_0	$(q_0, 1, R)$	$(q_0, 0, R)$	(q_1, B, R)
q_1	—	—	—

$M = (\{q_0, q_1\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_1\})$

2. Design Turing machine for 2's complement of the binary numbers.

Solution:

i/p 1 - 0101 o/p: 1011

δ	0	1	B
→ q_0	$(q_0, 1, R)$	$(q_0, 0, R)$	(q_1, B, L)
q_1	$(q_2, 1, L)$	$(q_1, 0, L)$	(q_2, B, R)
* q_2	—	—	—

$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_2\})$

Design Turing machine for left shift operation of the binary numbers.

δ	0	1	X	Y	B
→q_0	(q_1, X, R)	(q_1, Y, R)	E	E	(q_4, B, R)
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	E	E	(q_2, B, L)
q_2	$(q_2, 0, L)$	$(q_2, 1, L)$	$(q_3, 0, L)$	$(q_3, 1, L)$	E
q_3	E	E	(q_4, B, R)	(q_4, B, R)	E
*q_4	—	—	—	—	—

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Design Turing machine for addition of two unary numbers.

Solution:

	δ	0	1	B
→	q_0	(q_1, B, R)		
	q_1	$(q_1, 0, R)$	$(q_2, 0, R)$	
	q_2	$(q_2, 0, R)$		(q_3, B, R)
*	q_3	—	—	—

$M = (\{q_0, q_1, q_2, q_3\}, \{0\}, \{0, 1, B\}, \delta, q_0, B, \{q_3\})$

2. Design Turing machine for proper subtraction of two unary numbers

Solution:

	δ	0	1	B
→	q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, L)
	q_1	(q_2, B, L)	(q_4, B, R)	
	q_2	$(q_2, 0, L)$	$(q_2, 1, L)$	(q_3, B, R)
	q_3	(q_0, B, R)		
*	q_4	—	—	—

$M = \{\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \}$

gn Turing machine for multiplication of two unary numbers

δ	0	1	X	Y	B
→q_0	(q_1, X, R)				
q_1		$(q_1, 1, R)$			(q_4, B, R)
q_2		(q_3, B, L)	(q_2, X, L)		(q_2, B, L)
q_3		$(q_3, 1, L)$			
q_4					(q_4, B, L)
q_5		(q_6, X, R)		$(q_0, 1, R)$	$(q_7, 1, L)$
q_6		$(q_6, 1, R)$			(q_2, B, L)
q_7		$(q_7, 1, L)$		$(q_8, 1, L)$	
q_8		$(q_8, 1, L)$			(q_4, B, R)
*q_9					

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_9\})$

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Design Turing machine to compute n^2

Solution:

δ	0	1	X	Y	B
$\rightarrow q_0$		(q_1, X, R)			(q_6, B, R)
q_1		$(q_1, 1, R)$			(q_2, B, L)
q_2		$(q_2, 1, L)(q_3, X, R)$			
q_3		(q_4, X, R)			(q_6, B, L)
q_4		$(q_4, 1, R)$			$(q_5, 1, L)$
q_5		$(q_5, 1, R)$			(q_0, B, R)
$*q_6$					

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0, 1\},$

$\{0, 1, X, Y, B\}, \delta, q_0, B, \{q_6\})$

uring machine to compute n!

S	0	1	X	Y	B
→q_0		(q_1, X, R)			(q_5, B, R)
q_1		$(q_1, 1, R)$			(q_2, B, L)
q_2		(q_3, X, L)	(q_2, X, L)		
q_3		$(q_3, 1, L)$			(q_4, B, R)
q_4			$(q_0, 1, R)$		
*q_5					

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0, 1\}, \{0, 1, X, Y, B\}$

$\delta, q_0, B, \{q_5\})$

Viva- Questions.

1. Define a computable function.

Solution:

A computable function is a function for which there exists an algorithm (or Turing machine) that can produce the correct output for any valid input in a finite number of steps. It operates within the bounds of a well defined procedure, yielding output after processing the input.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Course Title	AUTOMATA THEORY AND FORMAL LANGAUGES	
Course Code(s)	22CS2002A	ACADEMIC YEAR: 2023-24

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 17:

Recursive and Recursively Enumerable Language

Concept Building

Recursive Languages

1. **Definition**:
   - A language is called **recursive** (or **decidable**) if there exists a Turing machine that will halt and accept for every string in the language, and halt and reject for every string not in the language.
2. **Characteristics**:
   - The Turing machine must always halt, regardless of whether the input string is in the language or not.
   - There exists an algorithm that can decide membership in the language.
3. **Examples**:
   - The set of all valid arithmetic expressions.
   - The set of all syntactically correct programs in a given programming language.

Recursively Enumerable Languages

1. **Definition**:
   - A language is called **recursively enumerable** (RE) if there exists a Turing machine that will halt and accept for every string in the language. However, for strings not in the language, the Turing machine may either halt and reject or run forever without halting.
2. **Characteristics**:
   - If a string is in the language, the Turing machine will eventually halt and accept it.
   - If a string is not in the language, the Turing machine may never halt.
   - The language can be "enumerated" by a Turing machine: there is a Turing machine that generates all strings in the language in some order, possibly with repetition.
3. **Examples**:
   - The set of all valid proofs in a formal system (like Peano arithmetic).
   - The set of all programs that halt for a given input (Halting problem instances).

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. **Write the difference between recursive and recursive enumerable language**

Solution:

- **Recursive**: The Turing machine always halts, and you can always determine if a string is part of the language.
- **Recursively Enumerable**: The Turing machine may halt for strings in the language but might run forever for strings outside the language.

2. **State the Decidability and Undecidability**

Solution:

Decidability and Undecidability:

- **Decidable**: A problem is decidable if a computer can always solve it in a finite time with a correct answer.
- **Undecidable**: A problem is undecidable if no algorithm can always solve it in a finite time, or it may run forever.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. List out the properties of Enumerable Languages

Properties of Enumerable Languages:

1. **Recognizable**: A Turing machine accepts strings in the language but may not halt for others.
2. **Semi-decidable**: Can decide membership for strings in the language, but not outside.
3. **Closure**:
    - Union, intersection, and concatenation of enumerable languages are enumerable.
4. **Not Closed Under Complement**: The complement may not be enumerable.
5. **Subset of Recursive**: All recursive languages are enumerable, but not all enumerable languages are recursive.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Write the characteristics of recursive and recursive enumerable language.

Solution:

Characteristics of Recursive Languages:

1. The Turing machine **always halts** for any input.
2. It **decides** whether a string belongs to the language.
3. Every recursive language is also **recursively enumerable**.
4. **Decidable** in finite time.

Characteristics of Recursively Enumerable Languages:

1. The Turing machine **may not halt** for strings not in the language.
2. It **recognizes** strings in the language.
3. The machine **accepts** strings in the language but might run indefinitely for others.
4. **Not closed** under complement.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Give an example of a recursive language and a recursively enumerable language?

Solution:

Example of a Recursive Language:

- **Even number of 1's**: The language $L = \{w \mid w \text{ contains an even number of 1's}\}$.

  A Turing machine can check the number of 1's and always halt with a decision (even or odd).

Example of a Recursively Enumerable Language:

- **Halting Problem**: The language $L = \{\langle M, w\rangle \mid M \text{ halts on input } w\}$.

  A Turing machine can accept the pair $\langle M, w\rangle$ if M halts on w, but it may not halt if M doesn't halt.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Briefly discuss about Properties of recursive and recursively enumerable languages.

Properties of Recursive Languages:

1. **Decidable**: Always halts and gives the correct answer.
2. **Closed**: Closed under union, intersection, complement.
3. **Subset**: All recursive languages are recursively enumerable.

Properties of Recursively Enumerable Languages:

1. **Semi-decidable**: May halt for strings in the language, but not for others.
2. **Closed**: Closed under union, intersection, but not complement.
3. **Subset**: Recursive languages are recursively enumerable.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

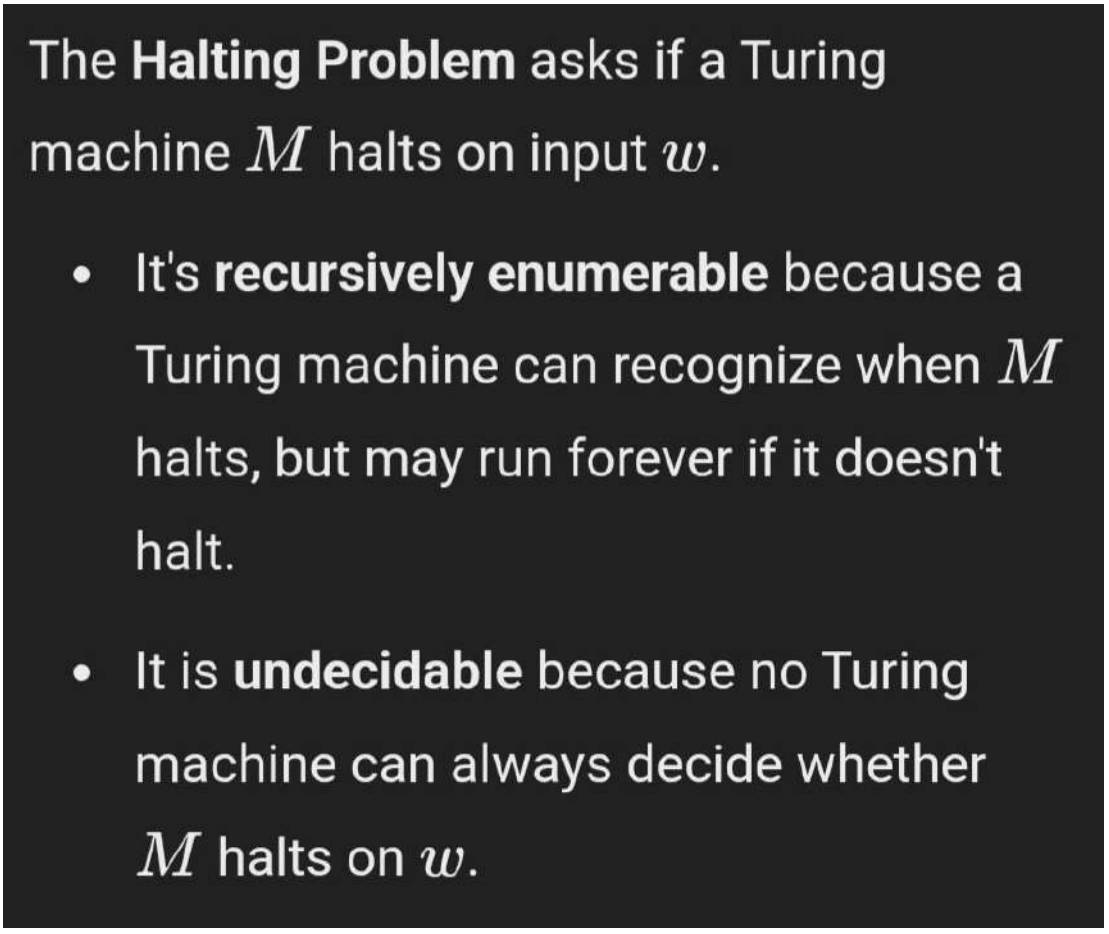
1. .How are recursive and recursively enumerable languages related?

Solution:

1. **Subset**: All **recursive** languages are **recursively enumerable**.
2. **Decidability**: Recursive languages are decidable, while recursively enumerable languages may not be.
3. **Complement**: The complement of a recursive language is recursive, but not necessarily for recursively enumerable languages.

2. explain the Halting problem in the context of recursively enumerable languages?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Why is a Turing machine called Linear Bounded Automata? Discuss the advantages of Linear Bounded Automata.

Solution:

A **Turing machine** is called a **Linear Bounded Automaton (LBA)** when its tape usage is restricted to a linear function of the input size.

Advantages of LBA:

1. **Space Efficiency**: Uses less memory compared to general Turing machines.
2. **Decidability**: Problems solvable by LBAs are decidable.
3. **Time Complexity**: Can often solve problems in polynomial time.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva – Questions

1. Give an example of a recursively enumerable language that is not recursive.

Solution:

The Halting Problem: $L = \{\langle M, w\rangle \mid M \text{ halts on } w\}$.

2. How are recursive languages related to the complexity class P?

Solution:

Recursive languages are decidable, and the complexity class P consists of problems solvable in polynomial time.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 18:

Universal Turing Machine

Concept Building

A Universal Turing Machine (UTM) is a powerful concept in computer science that embodies the idea of a single machine capable of simulating any other Turing machine. This concept is foundational for understanding computation and the limits of what can be computed.

Key Concepts of Universal Turing Machines

1. **Definition**: A Universal Turing Machine is a theoretical model that can execute any algorithm that can be described by a Turing machine. It does this by reading the description of another Turing machine from its input tape and simulating its behaviour.
2. **Encoding**: The UTM takes two inputs:
    - The description of a Turing machine (M), which includes its states, symbols, and transition functions.
    - An input string (w) that M will process.

The UTM uses this information to simulate the computation of M on the input w.

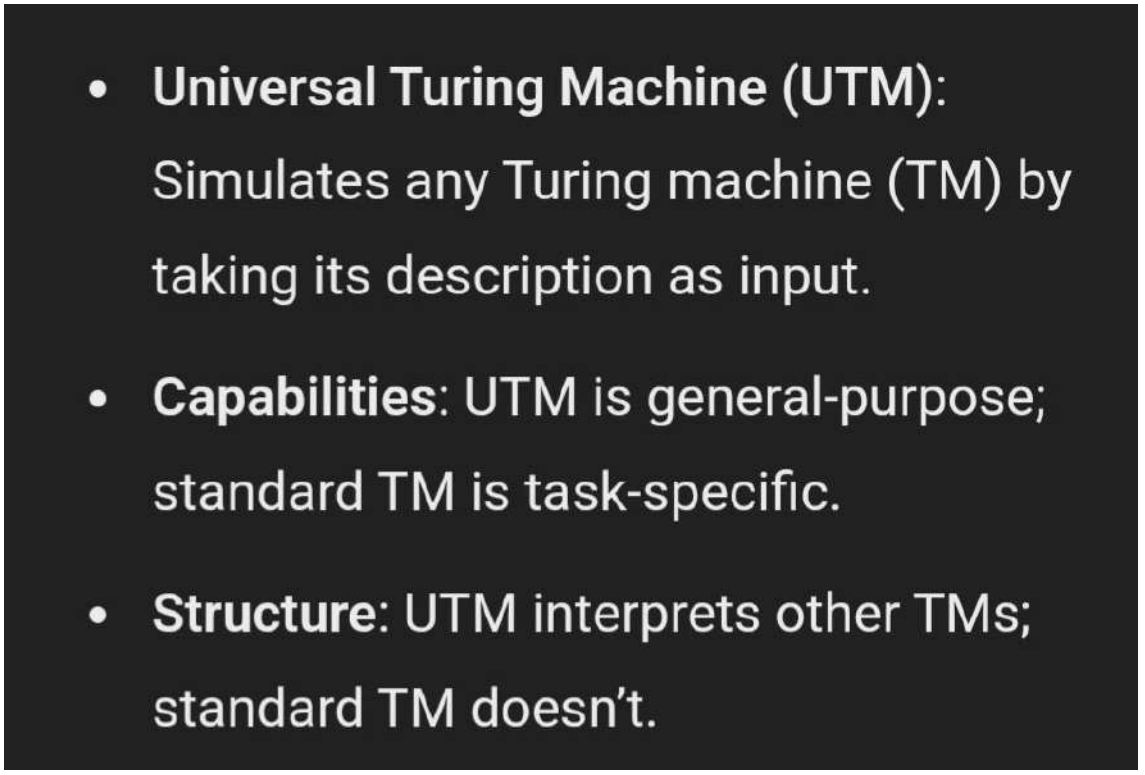
3. **Significance**: The existence of a UTM demonstrates that a single machine can perform any computation that can be algorithmically defined. This is a cornerstone of the Church-Turing thesis, which posits that any effectively calculable function can be computed by a Turing machine.
4. **Implications for Modern Computing**: The UTM concept laid the groundwork for the development of modern computers, which can execute a wide variety of programs. It illustrates the principle of a stored-program computer, where a machine can be reprogrammed to perform different tasks without changing its physical structure.
5. **Universality**: The UTM is universal in the sense that it can simulate any other Turing machine, regardless of its complexity. This means that any computation that can be performed by a specific Turing machine can also be performed by the UTM, albeit potentially less efficiently.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. What does it mean for a Turing machine to be universal? How does a UTM differ from a standard Turing machine in terms of its capabilities?

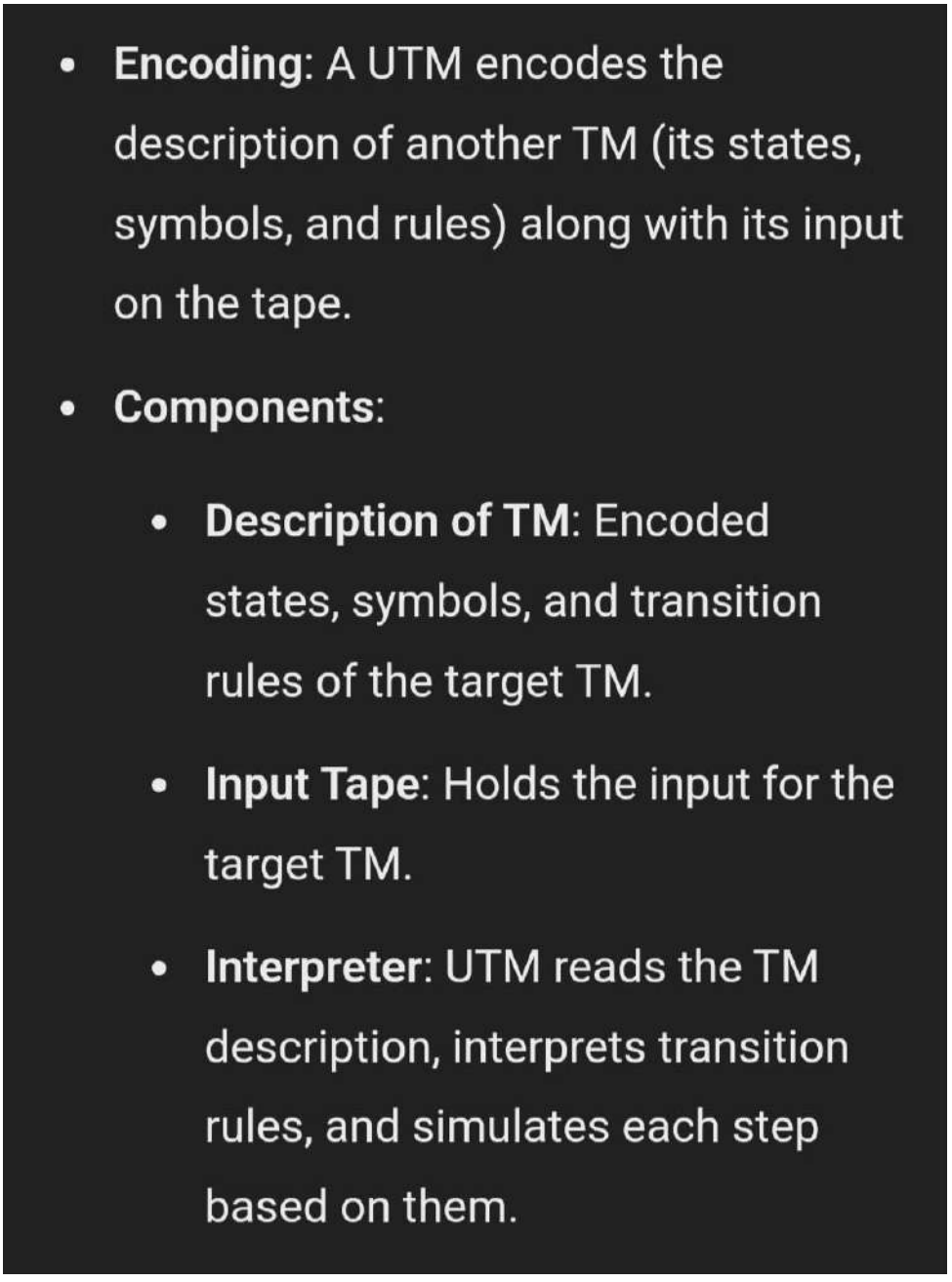
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. How does a UTM simulate the behavior of other Turing machines? What are the key components needed for this simulation?

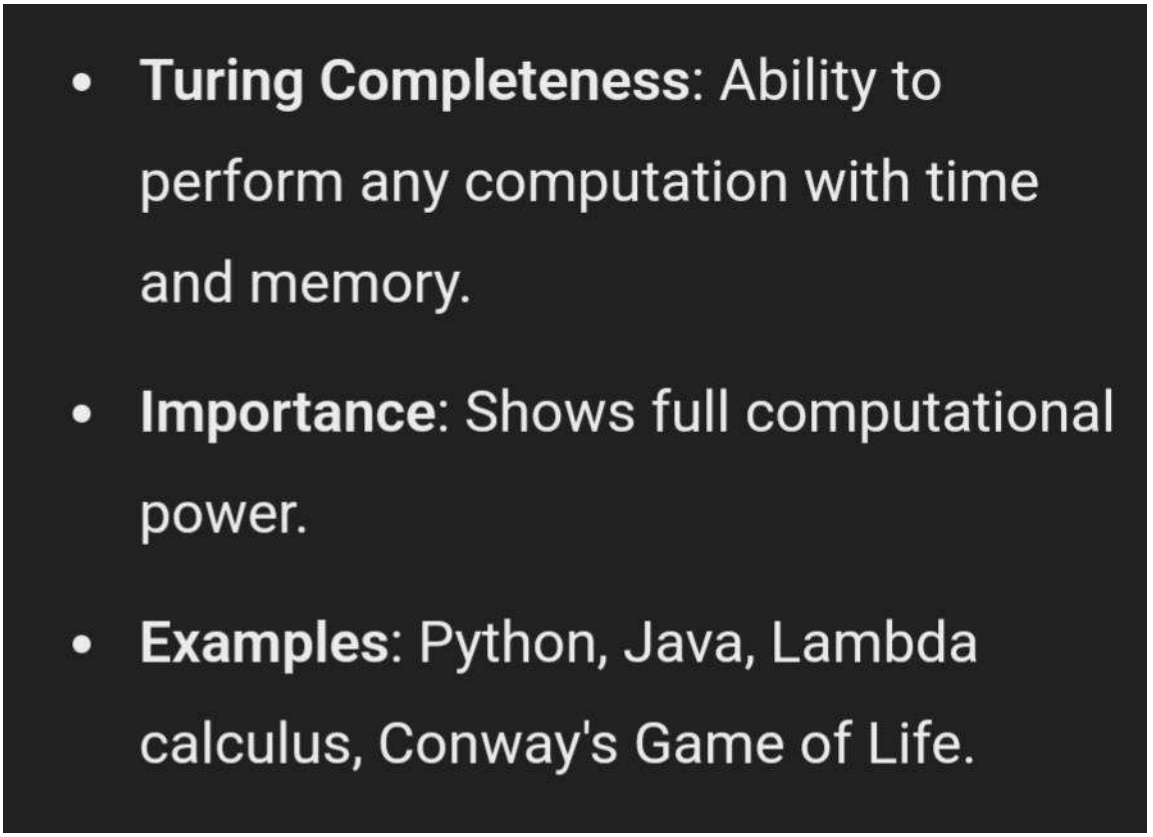
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. What is Turing completeness, and why is it an important concept in computer science? Can you provide examples of Turing complete systems?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

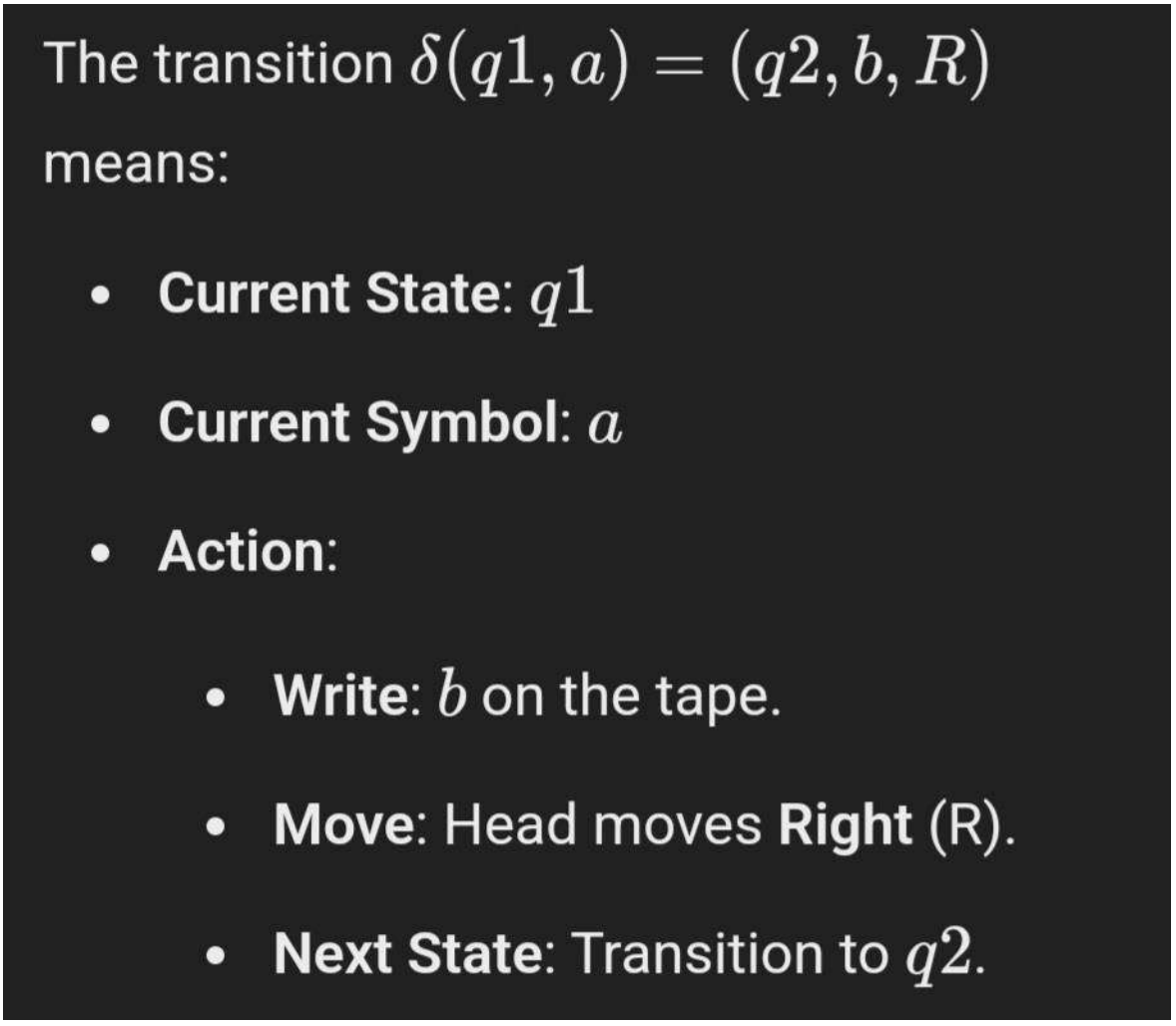
IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Consider a Universal Turing Machine that simulates another Turing machine M. If M has a transition defined as follows:

δ(q1, a) = (q2, b, R)

What does this transition mean?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. **How does a Universal Turing Machine utilize the description of another Turing machine?**

Solution:

A Universal Turing Machine (UTM) uses the description of another Turing machine (TM) as follows:

- **Encodes** the TM's states, symbols, and transition rules on its tape.
- **Interprets** these rules to simulate each step of the TM.
- **Executes** actions (write, move, state change) based on the encoded description, mimicking the TM's behavior.

3. **If a Universal Turing Machine simulates another Turing machine M, what must be included in the input to the UTM?**

Solution:

The input to a Universal Turing Machine (UTM) must include:

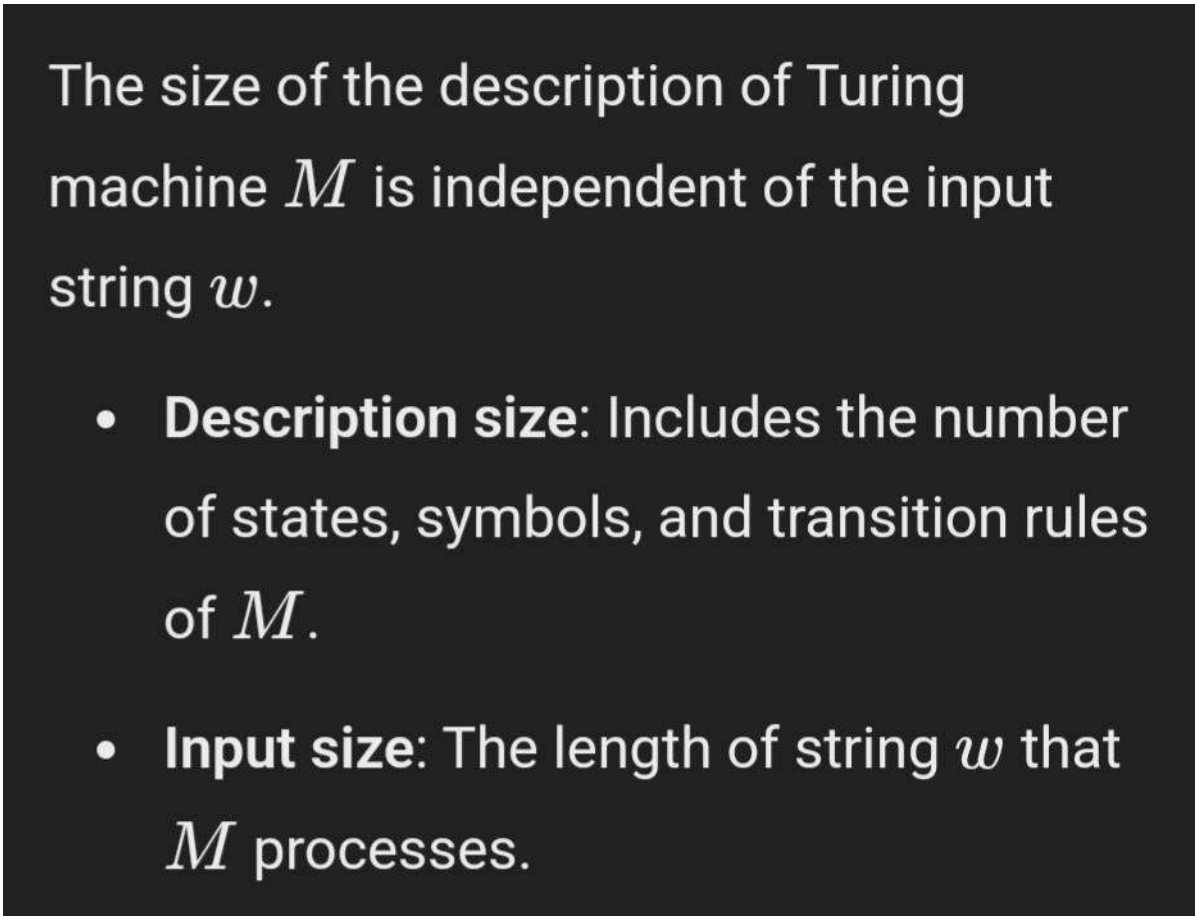
1. **Description of the Turing machine M**:
   - The states, symbols, and transition function of M.
2. **Input for M**:
   - The data or input that M is supposed to process.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

1. **How does the size of the description of a Turing machine M relate to the size of the input string w when simulating M on w using a Universal Turing Machine?**

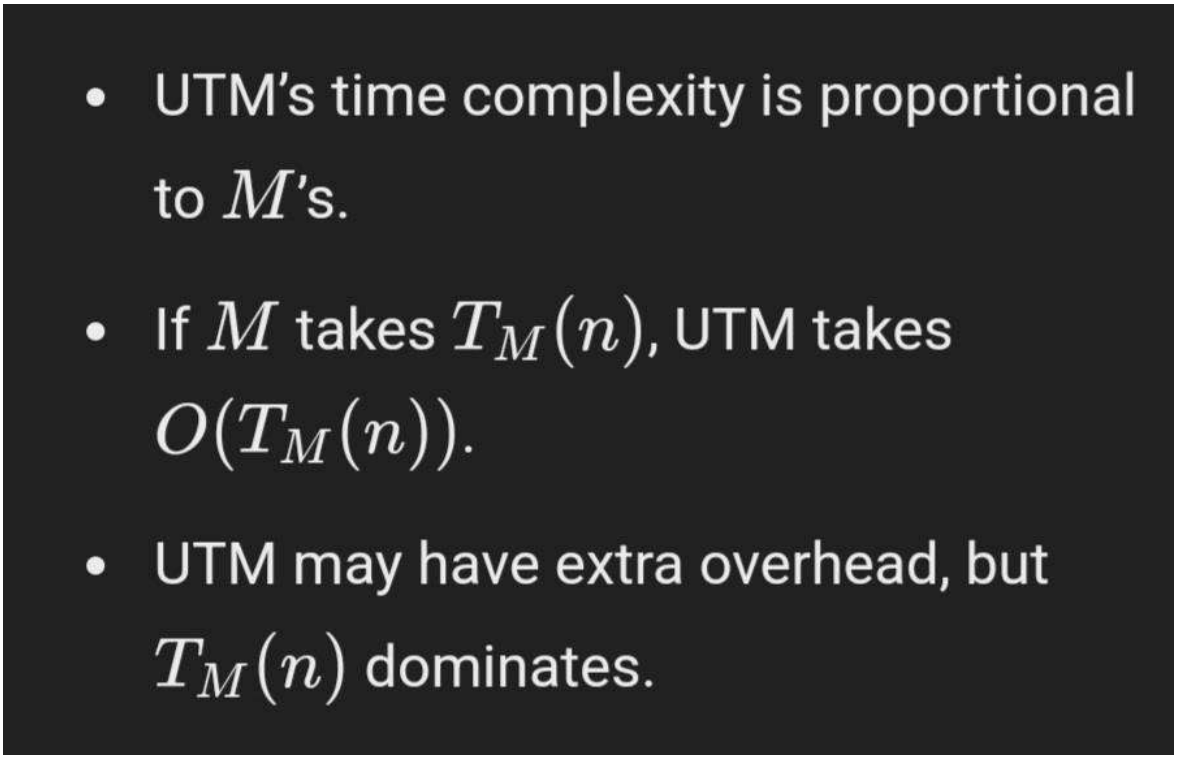
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. **When a Universal Turing Machine simulates another Turing machine M, what is the relationship between the time complexity of the UTM and the time complexity of M?**

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva – Questions

1. How does a Universal Turing Machine differ from a standard Turing machine?

Solution:

A UTM simulates any Turing machine, while a standard TM performs a specific task.

2. Explain how a Universal Turing Machine can simulate another Turing machine.

Solution:

A UTM encodes a Turing machine's description and input, then simulates its transitions step-by-step.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 19:

Chomsky Hierarchy and Halting Problem

Concept Building

Chomsky Hierarchy

The Chomsky hierarchy is a classification of formal grammars that describes the relationship between different types of languages and the computational power required to recognize them. It consists of four levels, each corresponding to a different class of languages and types of automata:

1. **Type 0: Unrestricted Grammars**
    - **Definition**: These grammars have no restrictions on their production rules. They can generate recursively enumerable languages.
    - **Equivalent Automaton**: Turing Machines (TMs).
    - **Example**: Any language that can be recognized by a Turing machine, such as the set of all strings over the alphabet {0, 1}.
2. **Type 1: Context-Sensitive Grammars**
    - **Definition**: These grammars have production rules of the form $\alpha A\beta \rightarrow \alpha w\beta$, where A is a non-terminal, and w is a string of terminals and non-terminals. The length of w must be greater than or equal to that of $\alpha A\beta$.
    - **Equivalent Automaton**: Linear Bounded Automata (LBA).
    - **Example**: The language $\{a^n b^n c^n \mid n \geq 1\}$.
3. **Type 2: Context-Free Grammars**
    - **Definition**: These grammars have production rules of the form $A \rightarrow \alpha A \rightarrow \alpha$, where AA is a single non-terminal and $\alpha\alpha$ is a string of terminals and non-terminals.
    - **Equivalent Automaton**: Pushdown Automata (PDA).
    - **Example**: The language $\{a^n b^n \mid n \geq 0\}$.
4. **Type 3: Regular Grammars**
    - **Definition**: These grammars have production rules of the form $A \rightarrow aB$ or $A \rightarrow a$, where A and B are non-terminals and aa is a terminal.
    - **Equivalent Automaton**: Finite Automata (FA).
    - **Example**: The language $\{a^* \mid n \geq 0\}$.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

The hierarchy is structured such that each level is a superset of the levels below it. For example, all regular languages (Type 3) are also context-free (Type 2), context-sensitive (Type 1), and recursively enumerable (Type 0). This hierarchy helps in understanding the computational complexity and capabilities of different types of languages and the machines that recognize them.

Halting Problem

The Halting Problem is a fundamental concept in computability theory that addresses the question of whether a given Turing machine will halt (i.e., finish its computation) on a given input or will run indefinitely.

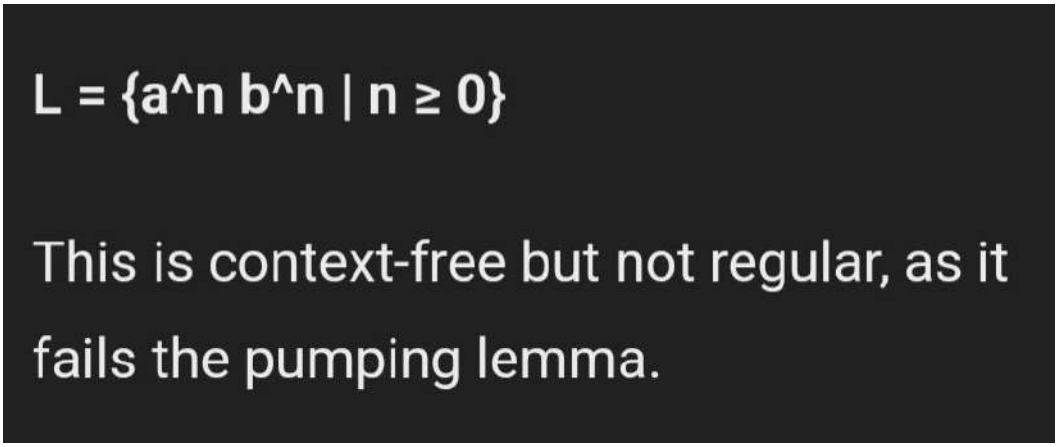
- **Definition**: The Halting Problem can be formally stated as follows: Given a Turing machine MM and an input string ww, determine whether MM halts when run with input ww.
- **Undecidability**: Alan Turing proved that there is no general algorithm that can solve the Halting Problem for all possible Turing machines and inputs. This means that it is impossible to construct a Turing machine (or any computational model) that can correctly determine whether any arbitrary Turing machine will halt on a given input.
- **Implications**: The undecidability of the Halting Problem has profound implications for computer science:
    - It establishes limits on what can be computed algorithmically.
    - It shows that certain problems cannot be solved by any algorithm, regardless of how powerful the machine is.
    - It leads to a deeper understanding of the boundaries between decidable and undecidable problems in computational theory.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

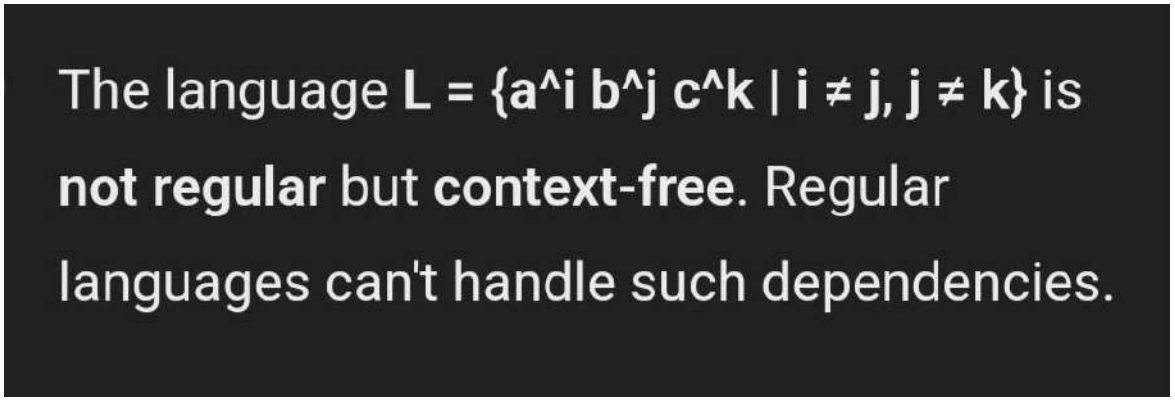
1. Give an example of a language that is context-free but not regular.

Solution:



2. Is the language L = {$a^i b^j c^k$ | i ≠j, j ≠ k} regular? If no, find the type of this langauge.

Solution:

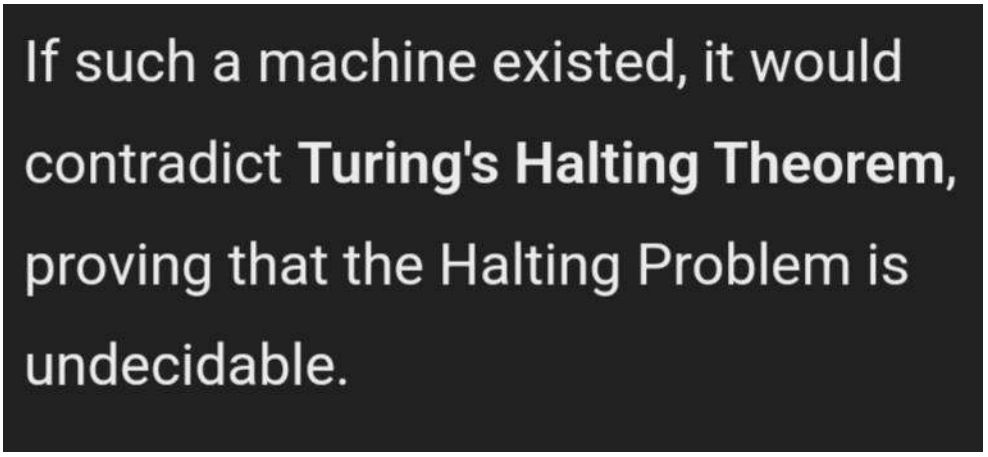


Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

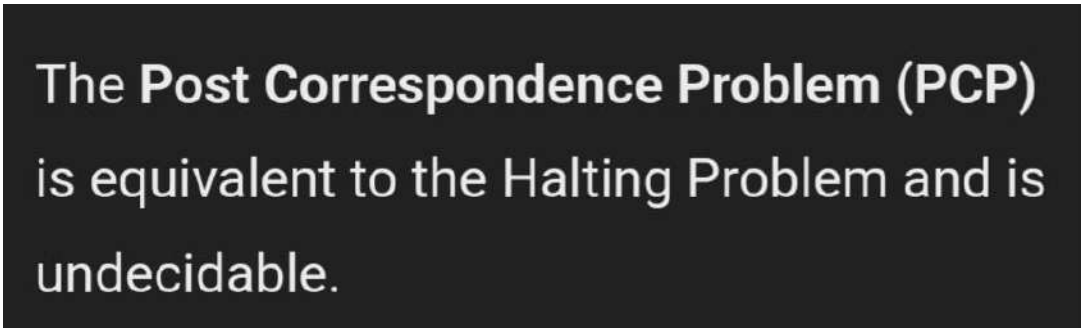
1. Consider a Turing machine that can determine whether another Turing machine halts on a given input. If such a machine exists, what can be the implications?

Solution:



2. Give a problem that is equivalent to halting problem.

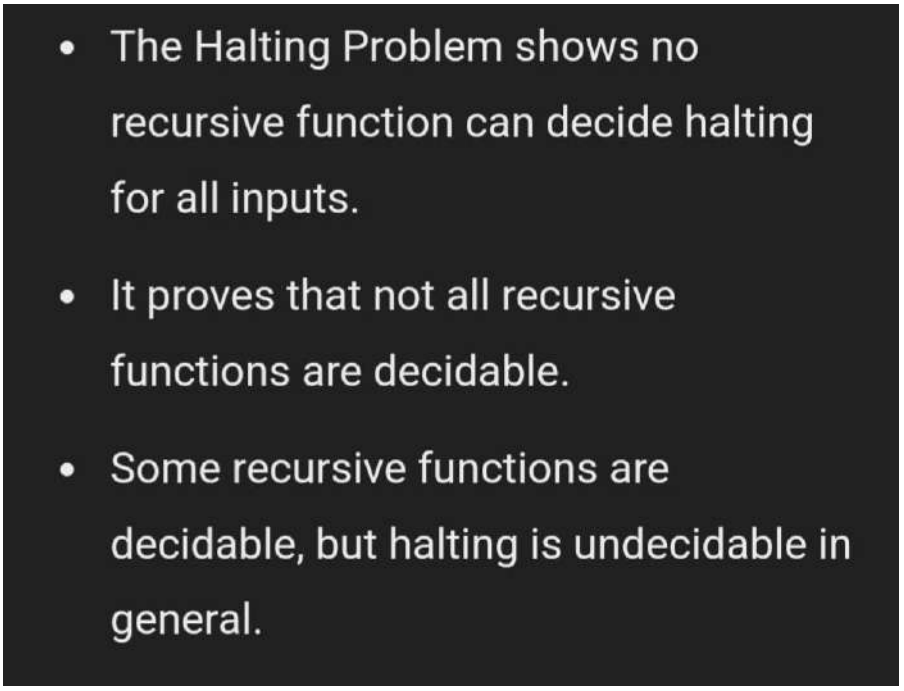
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

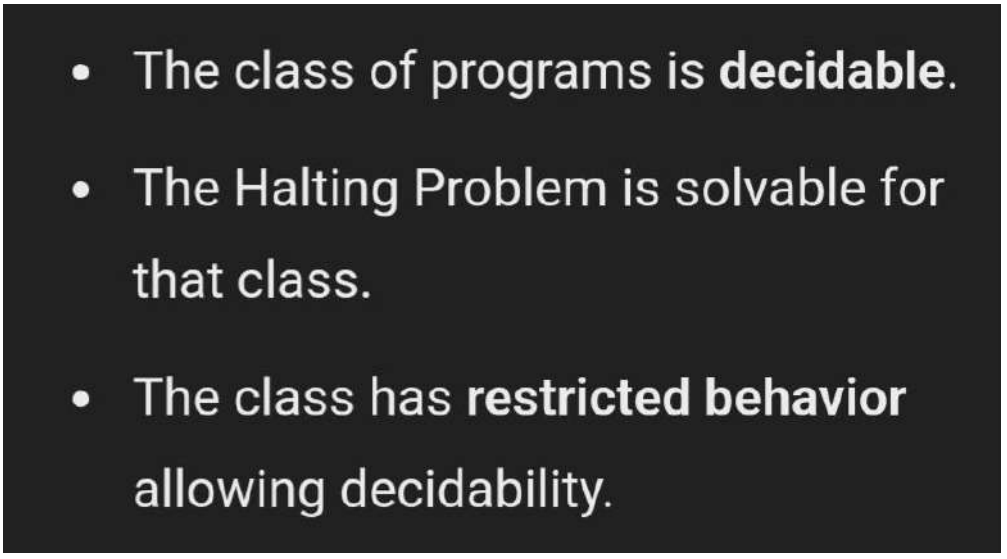
3. Describes the relationship between the Halting Problem and recursive functions.

Solution:



4. If we can create a Turing machine that solves the Halting Problem for a specific class of programs, what can we conclude?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

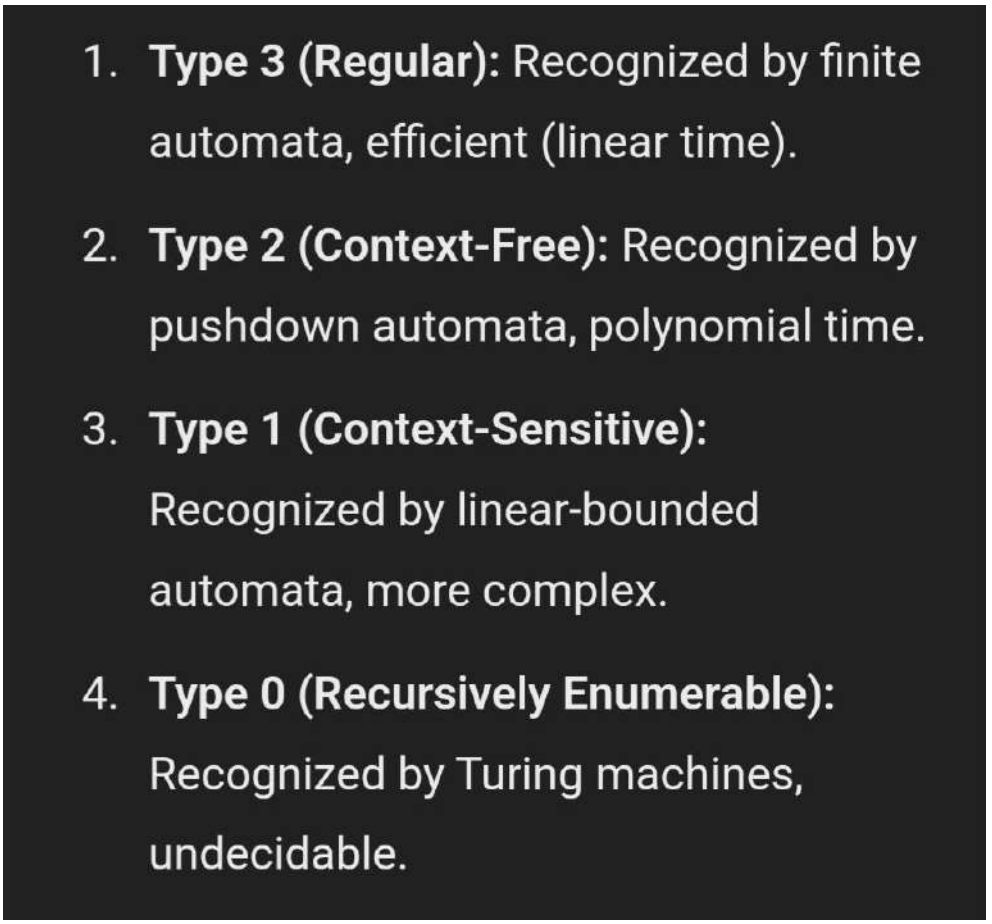
1. Compare and contrast Type 1 and Type 2 languages in terms of their generative grammars and recognizers.

Solution:

- **Type 1 (Context-Sensitive):** Generated by context-sensitive grammars, recognized by linear-bounded automata.
- **Type 2 (Context-Free):** Generated by context-free grammars, recognized by pushdown automata.
- **Comparison:** Type 1 is more powerful, with more complex recognition requirements than Type 2.

2. Illustrate the relationship between the Chomsky Hierarchy and computational complexity.

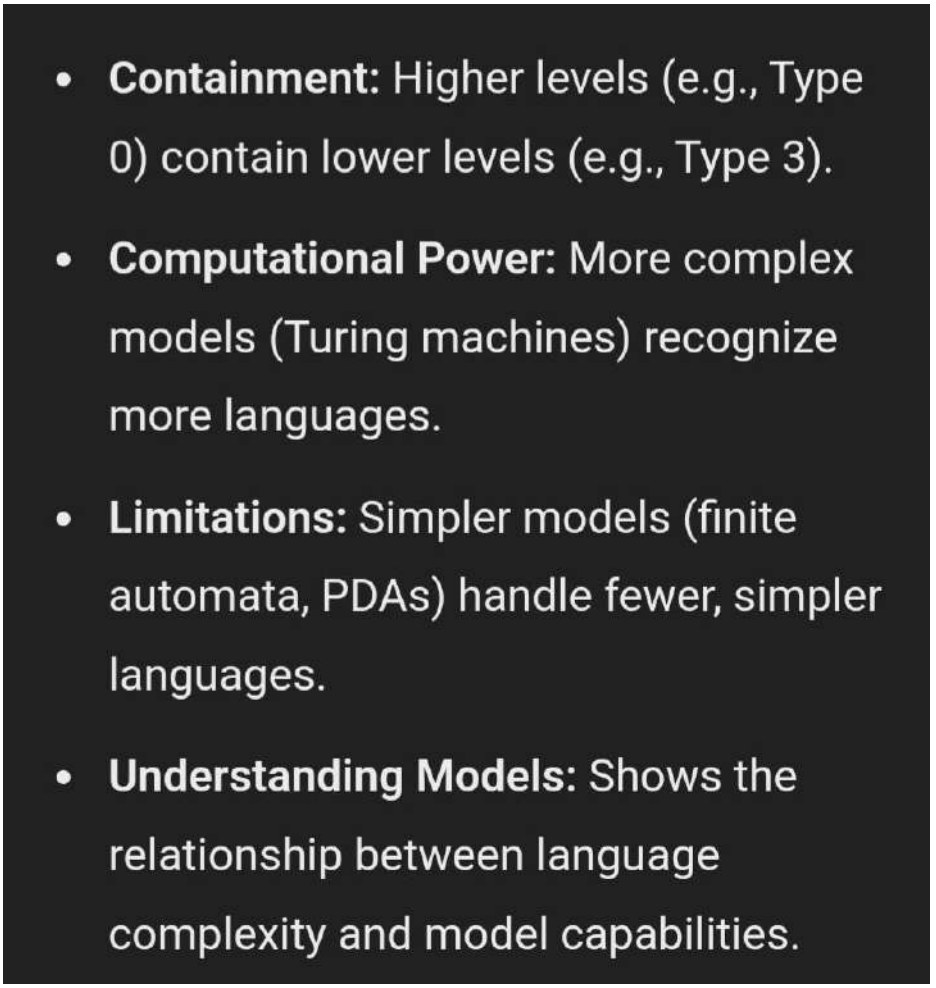
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. How does the concept of language containment within the Chomsky Hierarchy help in understanding the capabilities of different computational models?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva - Questions

1. Explain the characteristics of context-free grammars (CFGs) and the types of languages they generate.

Solution:

Context-free grammars generate **context-free languages**, with production rules of the form **A → γ**.

2. How do regular languages relate to context-free languages in the Chomsky hierarchy?

Solution:

Regular languages are a subset of context-free languages, with simpler rules and less computational power.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

CO - 4

TUTORIAL SESSION 20:

Variants of TM: Multi-Tape and NDTM

Concept Building

Multi-Tape Turing Machine (MTM)

A Multi-Tape Turing Machine (MTM) is an extension of the classical single-tape Turing machine. It consists of multiple tapes, each with its own head for reading and writing. Here are some key features:

Multiple Tapes: Instead of one tape, an MTM has several tapes, typically kkk tapes where kkk is a positive integer.

Independent Heads: Each tape has its own head that can move independently of the others, either left, right, or stay in place.

Transition Function: The transition function is more complex and takes into account the symbols under all kkk heads and dictates the new states, symbols to write, and movements of all heads.

Enhanced Computational Power: While MTMs do not provide more computational power in terms of solvable problems (they are equivalent to single-tape Turing machines in this respect), they can be more efficient, potentially solving problems faster by simulating certain computations more effectively.

Non-Deterministic Turing Machine (NDTM)

A Non-Deterministic Turing Machine (NDTM) is a theoretical model of computation that extends the classical deterministic Turing machine with the ability to explore multiple computation paths simultaneously. Here are the key features:

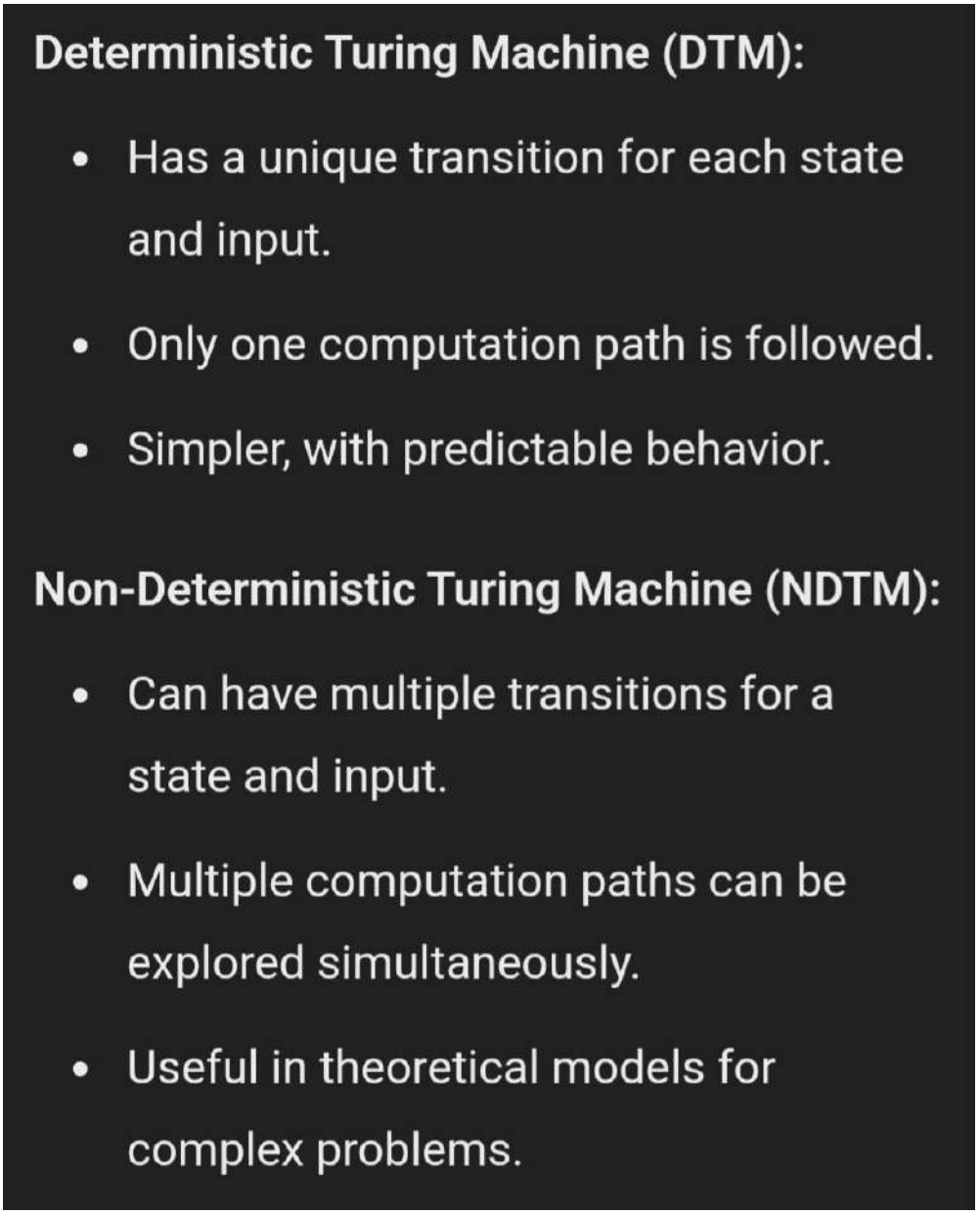
5. **Non-Deterministic Transitions:** At any given state and tape configuration, an NDTM can have multiple possible transitions.
6. **Parallel Computation Paths:** The machine can be thought of as exploring all possible transitions in parallel. If any computation path leads to an accepting state, the NDTM accepts the input.
7. **Acceptance Criteria:** An input is accepted if there exists at least one sequence of transitions leading to an accepting state.
8. **Theoretical Power:** NDTMs are used primarily in theoretical computer science to study problems in the complexity class NP (nondeterministic polynomial time). They are not realizable as physical machines but provide a useful abstraction for understanding computational complexity.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Differentiate deterministic and non-deterministic Turing Machine.

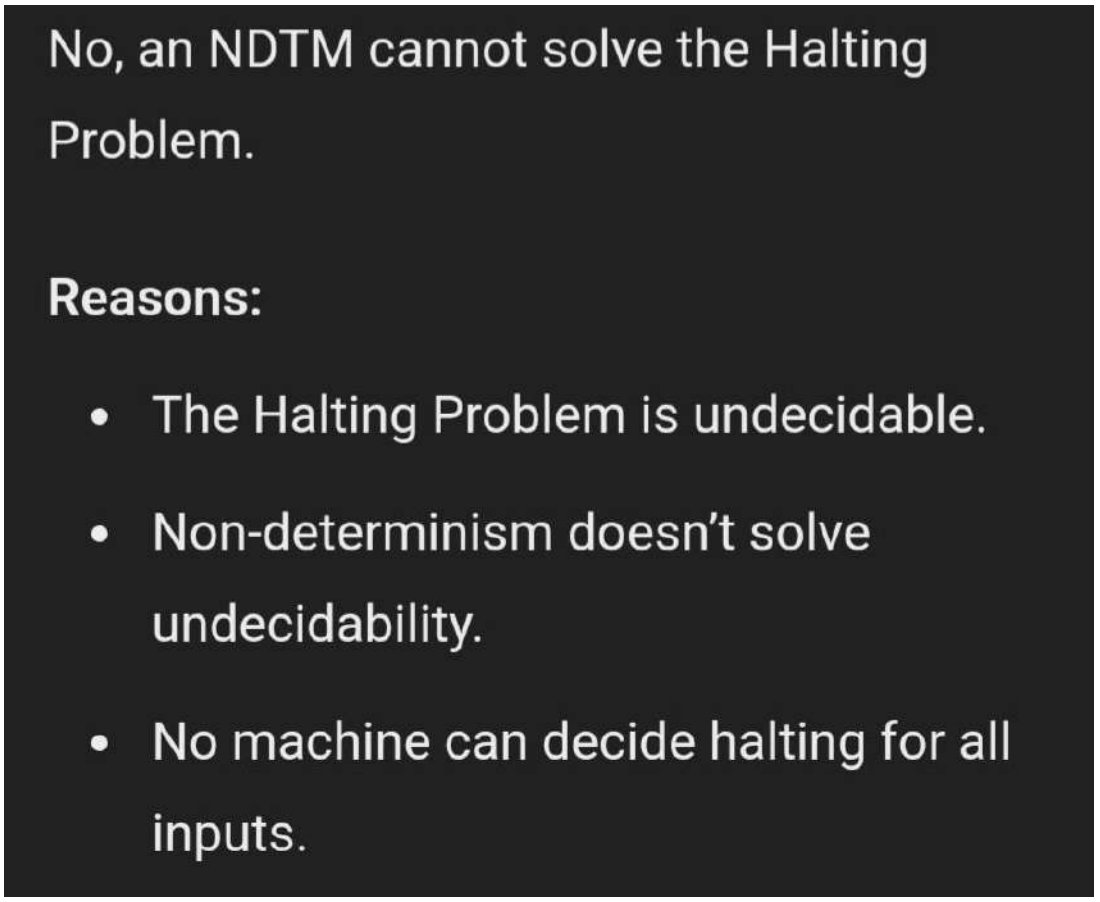
Solution:.



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Can a non-deterministic Turing machine be used to solve Halting Problem? Explain.

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Write the transition function for a k-tape Turing machine.

Solution:

$\delta(q, s_1, s_2, ..., s_k) = (q', r_1, r_2, ..., r_k, d_1, d_2, ..., d_k)$

where:

- **q**: current state.
- $s_1, s_2, ..., s_k$: symbols read on **k** tapes.
- **q'**: new state.
- $r_1, r_2, ..., r_k$: symbols to write.
- $d_1, d_2, ..., d_k$: move directions (L, R, S).

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. What is the primary advantage of a multi-tape Turing machine over a single-tape Turing machine?

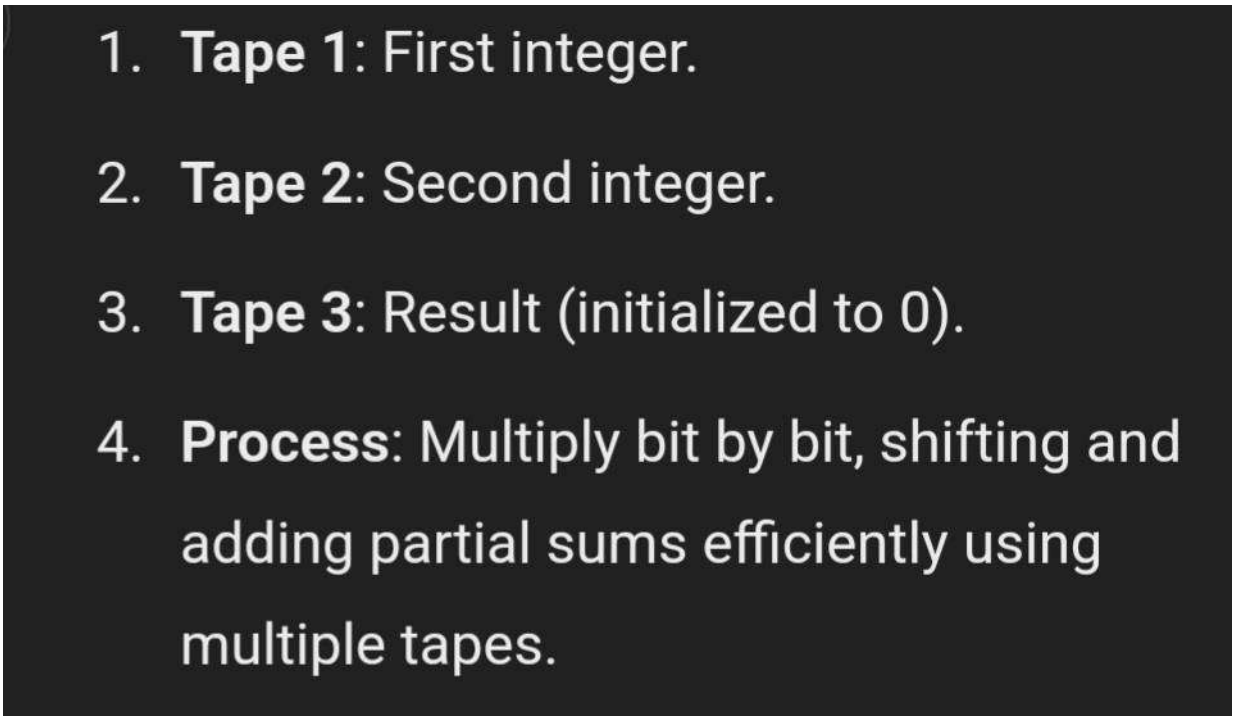
Solution:

- **Efficiency**: Simultaneous read/write on multiple tapes.
- **Reduced scanning**: Avoids repetitive tape movements.
- **Faster computation**: Complex tasks are completed more quickly.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

2. Explain how multiplication of two integers can be done using multi-tape Turing Machine.

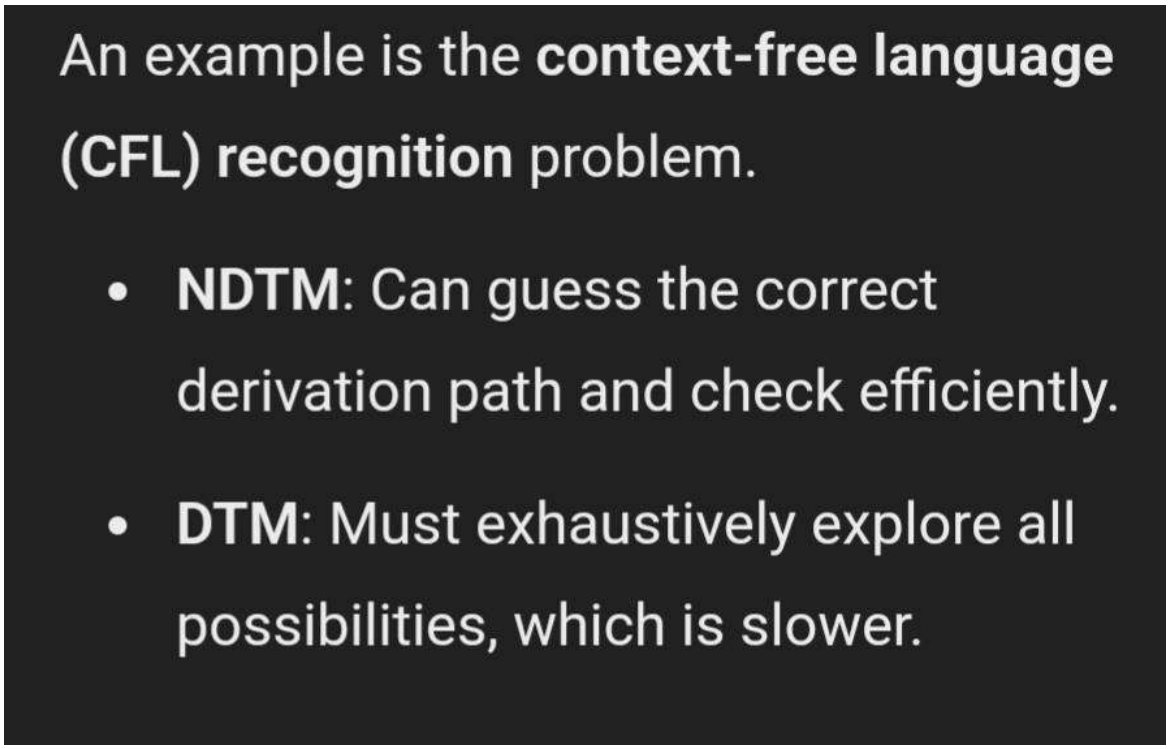
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. What is an example of a problem that is easier to solve with an NDTM?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

1. How is the time complexity of an NDTM measured?

Solution:

Time complexity of an NDTM is measured by the longest path it explores, assuming all paths are checked in parallel.

2. Can the behavior of an NDTM be simulated by a DTM? Explain with an example.

Solution:

Yes, an NDTM can be simulated by a DTM.

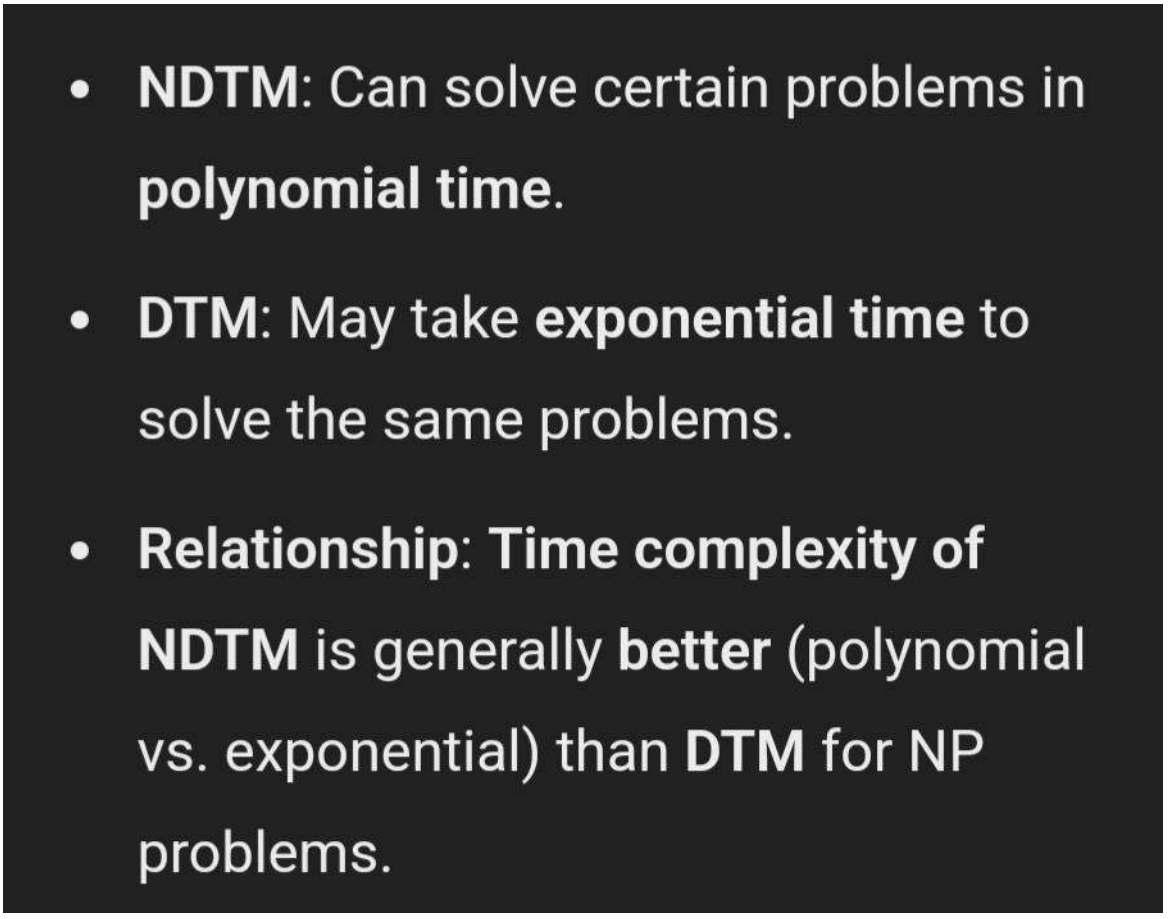
Example: For **3-SAT**:

- **NDTM**: Simultaneously tests all variable assignments.
- **DTM**: Checks each assignment one by one, leading to exponential time.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. What is the relationship between the time complexity of a non-deterministic Turing machine and a deterministic Turing machine?

Solution:

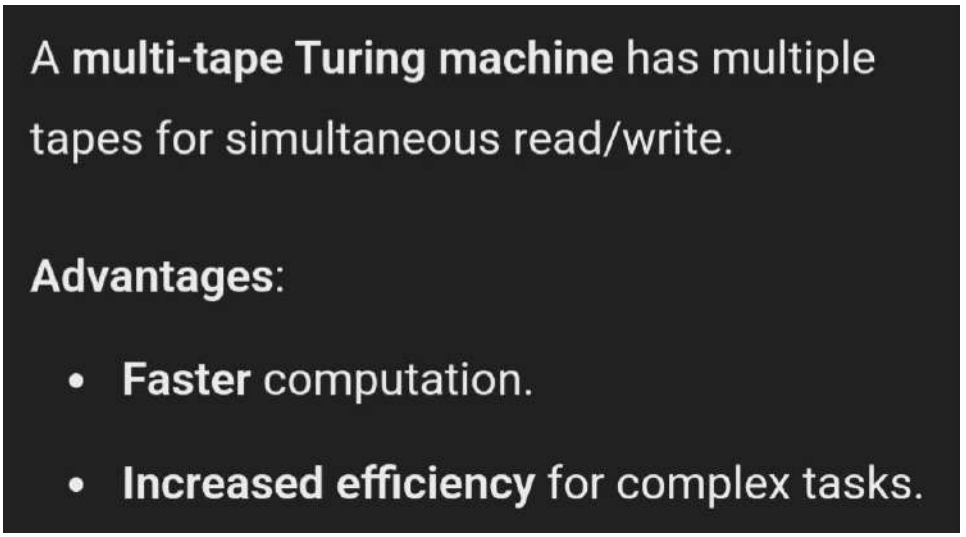


Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva – Questions

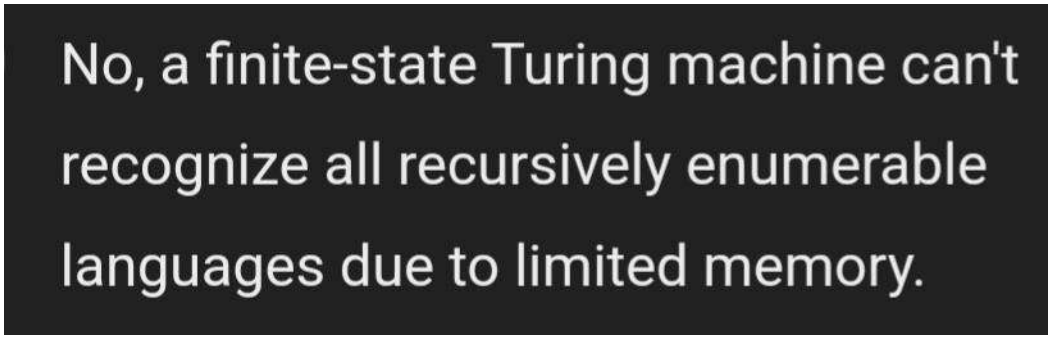
1. Explain the concept of a multi-tape Turing machine and its advantages over a single-tape Turing machine.

Solution:



2. Can a Turing machine with a finite number of states recognize all recursively enumerable languages? Explain.

Solution:



(For Evaluator's use only

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 21:

Time complexity of various problems

Concept Building

Time complexity is a critical concept in computer science that quantifies the amount of time an algorithm takes to complete as a function of the length of the input. It provides a way to analyze the efficiency of algorithms and helps in comparing different algorithms for the same problem.

Key Concepts of Time Complexity

1. **Definition**: Time complexity is expressed as a function of the size of the input, typically denoted as n. It describes how the runtime of an algorithm increases as the input size increases.
2. **Big O Notation**: The most common way to express time complexity is through Big O notation, which provides an upper bound on the time complexity of an algorithm. It describes the worst-case scenario of an algorithm's growth rate.
    - **Examples**:
        - O(1): Constant time – the algorithm's runtime does not change with the input size.
        - O(n): Linear time – the runtime increases linearly with the input size.
        - $O(n^2)$: Quadratic time – the runtime increases quadratically with the input size.
        - $O(2^n)$: Exponential time – the runtime doubles with each additional input element.
3. **Polynomial vs. Exponential Time**:
    - **Polynomial Time**: An algorithm is said to run in polynomial time if its time complexity can be expressed as $O(n^k)$ for some constant kk. Polynomial time is generally considered efficient and feasible for computation.
    - **Exponential Time**: An algorithm is said to run in exponential time if its time complexity can be expressed as $O(2^n)$ or similar. Exponential time algorithms are often impractical for large inputs due to their rapid growth.
4. **Classes of Problems**:
    - **P (Polynomial Time)**: The class of problems that can be solved by a deterministic Turing machine in polynomial time. Examples include sorting algorithms and searching algorithms.
    - **NP (Nondeterministic Polynomial Time)**: The class of decision problems for which a proposed solution can be verified in polynomial time. Examples include the Subset Sum problem and the Traveling Salesman problem.
    - **NP-Complete**: A subset of NP problems that are as hard as the hardest problems in NP. If any NP-complete problem can be solved in polynomial time, then all NP problems can be solved in polynomial time. Examples include the 3-SAT problem and the Hamiltonian Cycle problem.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

- **NP-Hard**: Problems that are at least as hard as NP-complete problems but do not need to be in NP. They may not even be decision problems. An example is the Halting Problem.

5. **Reduction**: This is a technique used to show that one problem is at least as hard as another. If problem A can be transformed into problem B in polynomial time, and if B is known to be hard, then A is also hard.

Examples of Time Complexity

1. **Linear Search**:
   - **Algorithm**: Search for an element in an unsorted list.
   - **Time Complexity**: O(n)– in the worst case, you may have to check every element.
2. **Binary Search**:
   - **Algorithm**: Search for an element in a sorted list.
   - **Time Complexity**: O(log n)– with each step, the search space is halved.
3. **Bubble Sort**:
   - **Algorithm**: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
   - **Time Complexity**: $O(n^2)$ – in the worst case, every element needs to be compared with every other element.
4. **Quick Sort**:
   - **Algorithm**: A divide-and-conquer algorithm that sorts by selecting a 'pivot' and partitioning the array into elements less than and greater than the pivot.
   - **Time Complexity**: Average case O(n * log n), worst case $O(n^2)$ (when the smallest or largest element is always chosen as the pivot).
5. **Traveling Salesman Problem (TSP)**:
   - **Algorithm**: Find the shortest possible route that visits each city exactly once and returns to the origin city.
   - **Time Complexity**: The brute-force solution is O(n!), which is exponential and impractical for large n.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Consider a non-deterministic Turing machine that decides a language L. If the time complexity of this NDTM is $O(n^k)$ for some constant k. Is L considered to be in P or NP? Explain.

Solution:

The language L is in **NP** because an NDTM decides it in polynomial time $O(n^k)$. However, we can't confirm L is in **P** without a deterministic polynomial-time algorithm.

2. **If a polynomial-time algorithm exists for any NP-complete problem, what can be concluded?**

Solution:

If a polynomial-time algorithm exists for any NP-complete problem, then **P = NP**.

3. What is the time complexity of the brute-force solution to the Subset Sum Problem?

Solution:

The time complexity is **O(2^n)**.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Give an example of a problem that is in NP but not known to be NP-complete?

Solution:

An example of a problem that is in **NP** but not known to be **NP-complete** is **Graph Isomorphism**.

2. Compare polynomial-time and exponential-time algorithms with examples.

Solution:

Type	Definition	Example Problem	Time Complexity
Polynomial-time	Solves in $O(n^k)$ for some constant k	Sorting (e.g., Merge Sort)	$O(n \log n)$
Exponential-time	Solves in $O(2^n)$ or similar large growth	Subset Sum (brute-force)	$O(2^n)$

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. What is the time complexity of Dijkstra's algorithm when implemented with a priority queue?

Solution:

The time complexity of Dijkstra's algorithm with a priority queue is **O((V + E) log V)**.

4. What is the time complexity of the DFS algorithm for a graph represented as an adjacency list?

Solution:

The time complexity of DFS using an adjacency list is **O(V + E)**.

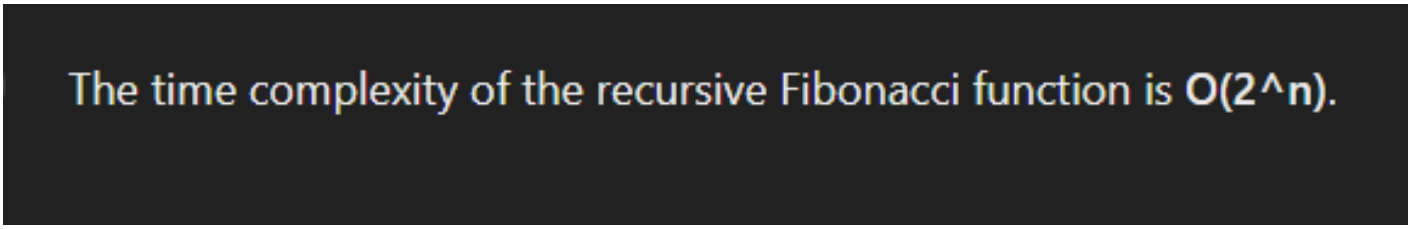
Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Determine the time complexity of the following recursive function:

```
function fib(n):
  if n <= 1:
    return n
  else:
    return fib(n-1) + fib(n-2)
```

Solution:

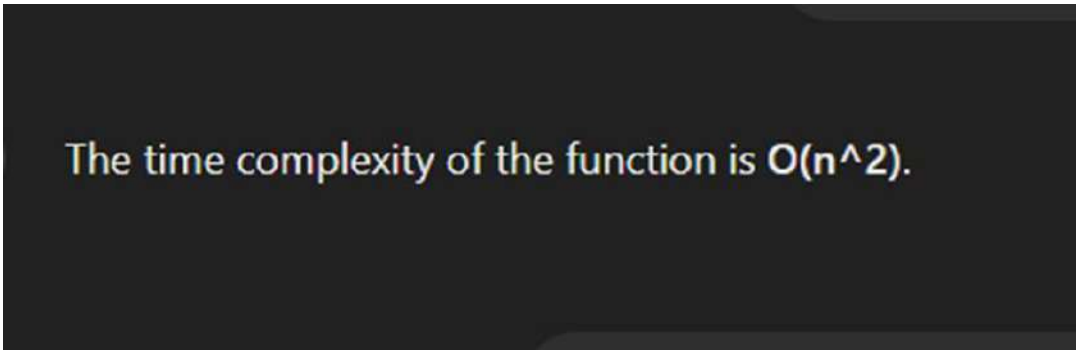


2. Determine the time complexity of the following function:

```
def example_function(n):
for i in range(n):
for j in range(n):
print(i, j)
```

Solution:

•



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Discuss the exponential time complexity of the Traveling Salesman Problem (TSP) using the brute force approach.

Solution:

The Traveling Salesman Problem (TSP) has a brute force time complexity of **O(n!)**. This is due to evaluating all permutations of cities, leading to impractical computation times as n increases.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva – Questions

1. What is time complexity, and why is it important in algorithm analysis?

Solution:

Time complexity measures an algorithm's efficiency as input grows.

2. Explain the difference between worst-case, average-case, and best-case time complexity.

Solution:

. Worst-case: maximum time, average-case: expected time, best-case: minimum time.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 22:

Class P, class NP, NP-Hard, NP-Complete

Concept Building

The concepts of **P** and **NP** are fundamental in computational complexity theory, which studies the resources required to solve computational problems. Here's a breakdown of these classes along with examples for better understanding.

What is P?

- **Definition**: The class **P** consists of decision problems (problems with a yes/no answer) that can be solved by a deterministic Turing machine in polynomial time. This means that there exists an algorithm that can solve the problem in time that can be expressed as a polynomial function of the input size nn.
- **Examples**:
    1. **Sorting**: Sorting a list of numbers can be done in O(nlog⁡n)O(nlogn) time using efficient algorithms like Merge Sort or Quick Sort.
    2. **Finding the Greatest Common Divisor (GCD)**: The Euclidean algorithm can compute the GCD of two numbers in O(log⁡(min⁡(a,b)))O(log(min(a,b))) time.
    3. **Graph Traversal**: Problems like finding the shortest path in a graph (using Dijkstra's or Bellman-Ford algorithms) can be solved in polynomial time.

What is NP?

- **Definition**: The class **NP** consists of decision problems for which a proposed solution can be verified in polynomial time by a deterministic Turing machine. In other words, if you are given a "certificate" (a proposed solution), you can check whether it is correct in polynomial time.
- **Examples**:
    1. **Subset Sum Problem**: Given a set of integers, is there a subset whose sum equals a given target? If someone provides a subset, you can easily sum its elements and check against the target.
    2. **Hamiltonian Cycle Problem**: Given a graph, does there exist a cycle that visits each vertex exactly once? If you are given a specific cycle, you can verify by checking if it visits all vertices and returns to the starting point.
    3. **Graph Coloring**: Given a graph, can you color the vertices with kk colors such that no two adjacent vertices share the same color? If a coloring is provided, you can check it in polynomial time.

Relationship Between P and NP

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

- **P vs NP**: The major question in computer science is whether P = NP. This means asking if every problem for which a solution can be verified quickly (in polynomial time) can also be solved quickly (in polynomial time).
- **NP-Complete**: A problem is NP-complete if it is in NP and as hard as any problem in NP. If any NP-complete problem can be solved in polynomial time, then every problem in NP can be solved in polynomial time (thus P = NP). Examples include:
    - **Traveling Salesman Problem** (decision version)
    - **Knapsack Problem**
    - **3-SAT Problem**

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Define the complexity classes P, NP, and NP-complete? Draw Venn diagram to show their relationship.

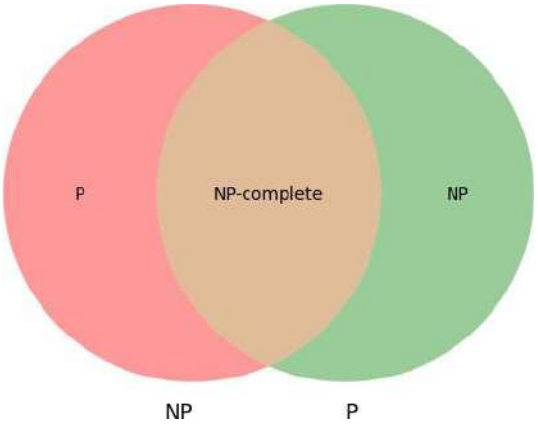
Solution:

Complexity Classes:

- **P**: Solvable in polynomial time (e.g., sorting).
- **NP**: Verifiable in polynomial time (e.g., SAT).
- **NP-complete**: Hardest problems in NP (e.g., SAT, TSP).

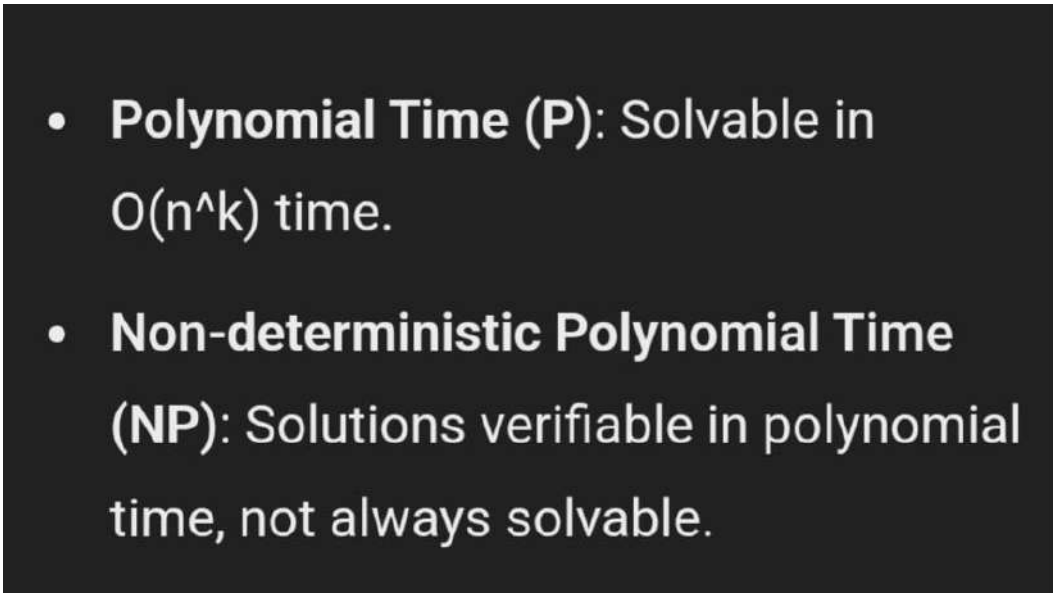
Venn Diagram:

- **P ⊆ NP**
- **NP-complete ⊆ NP** but not necessarily in **P**.



2. Discuss the difference between polynomial time and non-deterministic polynomial time.?

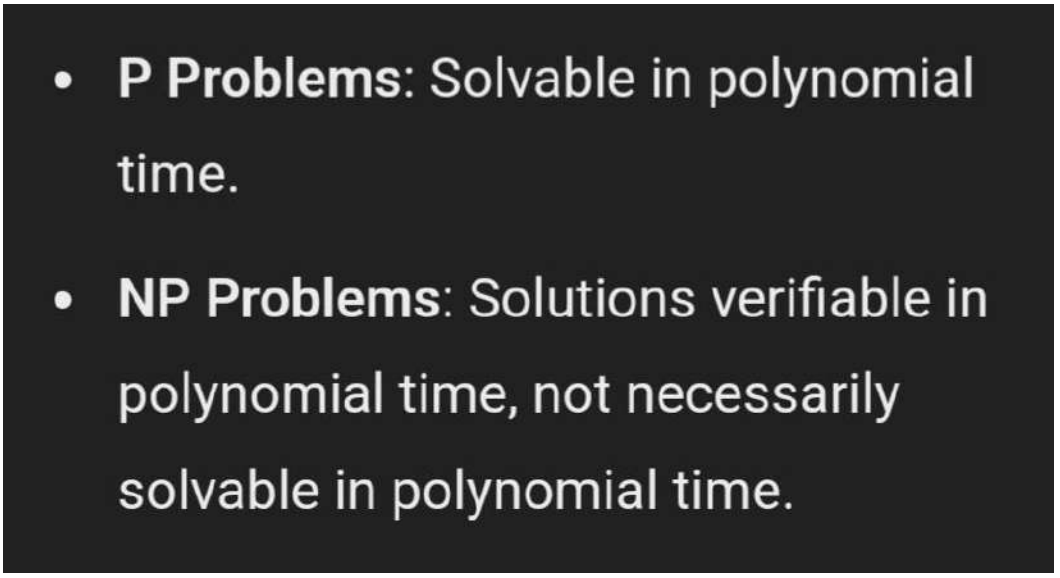
Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Mention the difference between P and NP problems?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Explain the concept of polynomial time complexity and its significance in the context of the P versus NP problem. How does the ability to solve problems in polynomial time differ between P and NP?

Solution:

- **Polynomial Time Complexity**: Solving in O(n^k) time.
- **P Problems**: Solvable in polynomial time.
- **NP Problems**: Verifiable in polynomial time, not necessarily solvable in polynomial time.
- **P vs NP**: Asks if every problem verifiable in polynomial time (NP) can also be solved in polynomial time (P). If P = NP, all NP problems are solvable efficiently.

2. What does it mean if a problem is NP-hard?

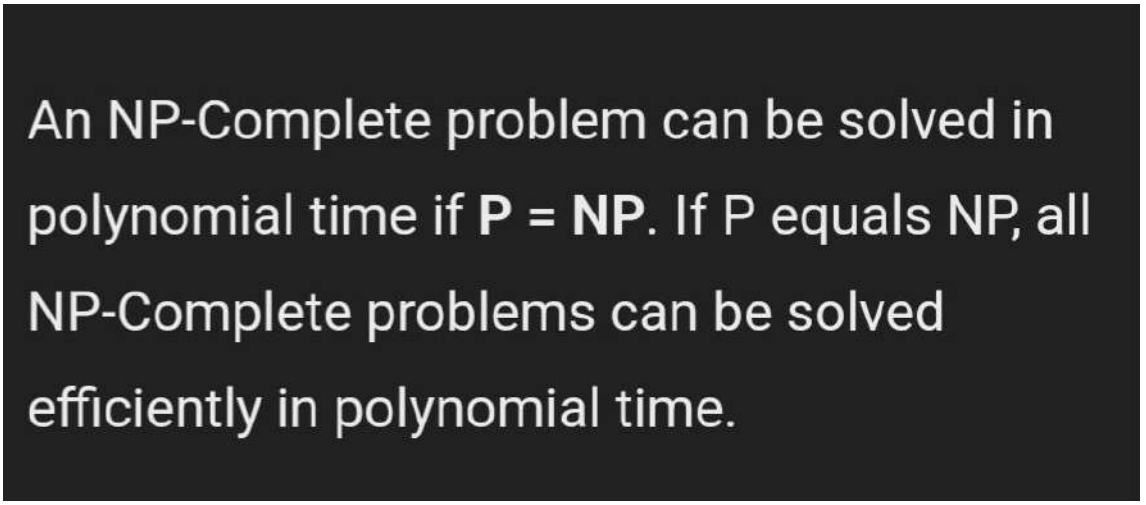
Solution:

NP-hard: A problem as hard as the hardest NP problems. Solving it efficiently would solve all NP problems efficiently.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

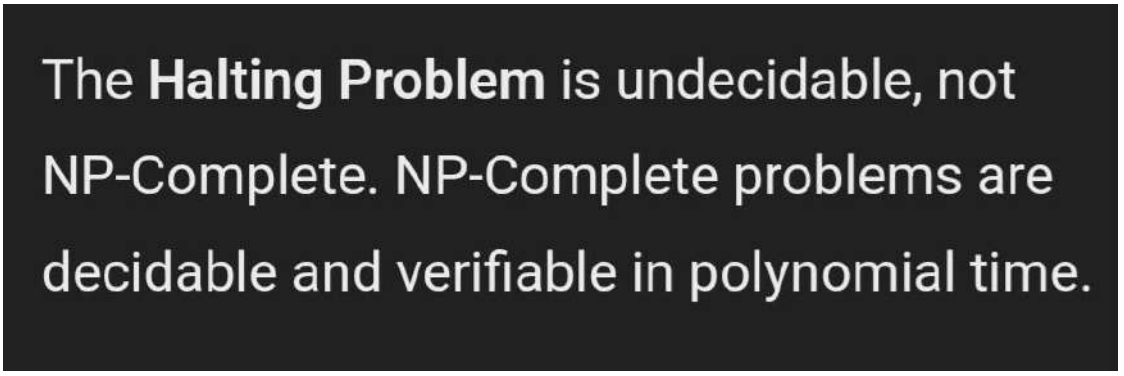
3. when a NP-Complete problem be solved in polynomial time?

Solution:



4. Can we say Halting problem is NP-Complete?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Define the characteristics of NP-Complete problems.

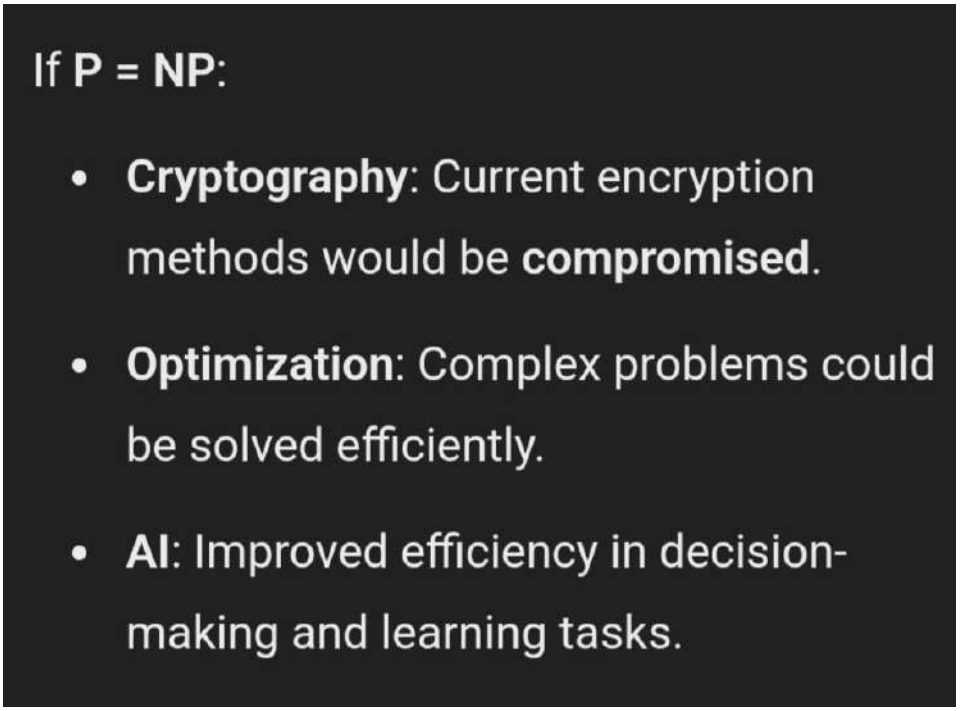
Solution:

Characteristics of NP-Complete Problems:

- **In NP**: Solutions can be verified in polynomial time.
- **NP-hard**: At least as hard as the hardest problems in NP.
- **Polynomial-time reduction**: Every NP problem can be reduced to it in polynomial time.
- **Decidable**: The problem has a solution, but finding it may take exponential time unless P = NP.

2. Describe the implications of proving P = NP. How would this breakthrough affect various fields, such as cryptography, optimization, and artificial intelligence?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Show that Kruskal's algorithm is in class P .?

Solution:

Kruskal's algorithm is in **P** because it runs in polynomial time:

- **Sorting edges**: $O(E \log E)$
- **Union-Find operations**: $O(\alpha(V))$

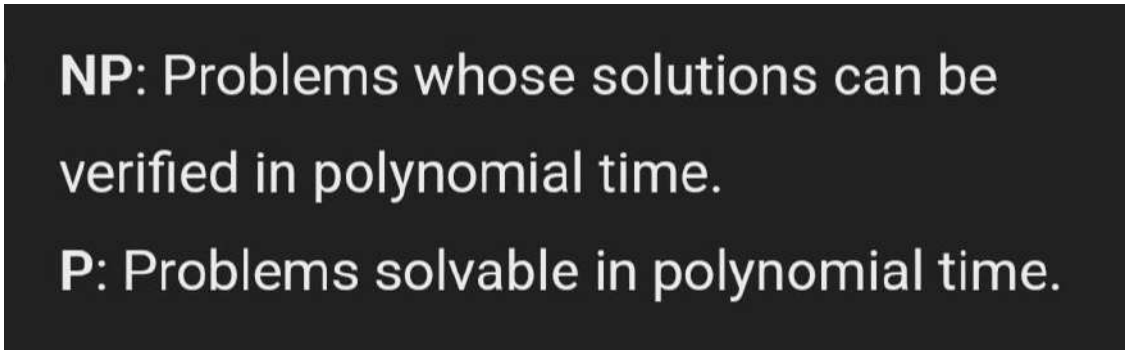
Thus, Kruskal's algorithm has a time complexity of $O(E \log E)$, which is polynomial, placing it in class **P**.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva - Questions

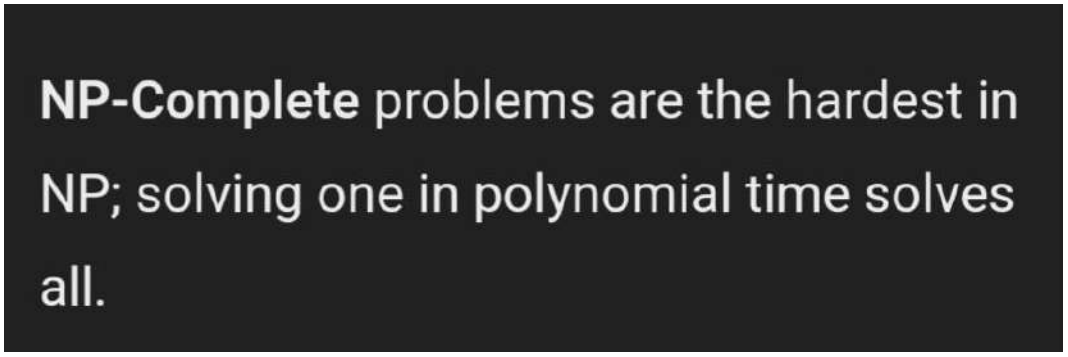
1. Explain the class NP and how it differs from class P.

Solution:



2. What are NP-complete problems, and why are they significant?

Solution:



(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 23:

PSPACE

Concept Building

PSPACE is a complexity class that represents the set of decision problems that can be solved by a Turing machine using a polynomial amount of space. This means that the amount of memory used by the algorithm grows polynomially with the size of the input, regardless of the time it takes to compute the solution.

Key Characteristics of PSPACE

1. **Definition**: A decision problem is in PSPACE if there exists an algorithm that can solve the problem using a polynomial amount of memory space. The time complexity can be exponential or even worse, but the space used must be polynomial.
2. **Relation to Other Complexity Classes**:
    - **P**: The class of problems that can be solved in polynomial time. It is a subset of PSPACE, meaning every problem in P can also be solved in PSPACE.
    - **NP**: The class of problems for which a solution can be verified in polynomial time. It is not known whether NP is a subset of PSPACE, but it is widely believed that PSPACE contains NP.
    - **PSPACE-complete**: A subset of PSPACE that contains the hardest problems in PSPACE. If any PSPACE-complete problem can be solved in polynomial time, then all problems in PSPACE can also be solved in polynomial time.
3. **Examples of PSPACE Problems**:
    - **Quantified Boolean Formula (QBF)**: This problem involves determining the truth of a quantified Boolean formula and is PSPACE-complete.
    - **Generalized Chess**: Determining the winner of a game of chess given any arbitrary position is PSPACE-complete.
    - **Tiling Problem**: The problem of determining whether a given set of tiles can tile a given area is PSPACE-complete.
4. **Space Complexity**: The space complexity of an algorithm is defined as the amount of memory it requires to solve a problem as a function of the input size. For example, an algorithm that uses space proportional to the square of the input size would have a space complexity of $O(n^2)$.
5. **PSPACE vs. Exponential Time**: While PSPACE allows for polynomial space usage, it does not limit the time complexity. Problems in PSPACE can take exponential time to solve, but they will not exceed polynomial space usage.
6. **Containment Relationships**:
    - $P \subseteq NP \subseteq PSPACE$

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

- PSPACE is believed to be strictly larger than NPNP, meaning there are problems in PSPACE that are not in NP.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Give an example of PSACE-Complete problem. Mention the reason for the same.

Solution:

Example: Quantified Boolean Formula (QBF)

Reason:

- Formula: $\exists x\ \forall y\ (x \vee \neg y)$.
- QBF uses polynomial space.
- Every PSPACE problem reduces to QBF in polynomial time.

2. What is the significance of Savitch's theorem in the context of PSPACE?

Solution:

Significance of Savitch's Theorem:

- **Formula**: $\mathrm{NPSPACE} = \mathrm{PSPACE}$
- **Implication**: Deterministic and nondeterministic polynomial space are equivalent.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. If a polynomial-space algorithm is found for one PSPACE-complete problem, what can be concluded?

Solution:

Solution:
If a polynomial-space algorithm is found for one PSPACE-complete problem, it implies that **all PSPACE-complete problems** and **all problems in PSPACE** can be solved in polynomial space, as they are all reducible to each other.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Give problems is known to be in PSPACE but not known to be PSPACE-complete?

Solution:

Problem: Tarski's Circle-Squaring Problem

Reason:

- Known to be in **PSPACE**.
- Not proven **PSPACE-complete** or **outside** PSPACE-complete boundaries yet.

2. What are the closure properties of PSPACE

Solution:

PSPACE Closure Properties:

1. **Complement**: Closed.
2. **Union**: Closed.
3. **Intersection**: Closed.
4. **Concatenation**: Closed.
5. **Polynomial-time reduction**: Closed.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Can you judge all PSPACE problems can be reduced to PSPACE Complete Problems

Solution:

Yes, all **PSPACE problems** can be reduced to **PSPACE-complete problems**. This is because **PSPACE-complete** problems are the hardest problems in PSPACE, and any problem in PSPACE can be polynomial-time reduced to a PSPACE-complete problem.

4. How can we measure the Polynomial Space.

Solution:

Polynomial space is measured by the memory usage $O(n^k)$, where k is a constant.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Write the relationship between PSPACE and EXPSPACE.

Solution:

Relationship between PSPACE and EXPSPACE:

- **PSPACE**: Solvable in polynomial space, $O(n^k)$.
- **EXPSPACE**: Solvable in exponential space, $O(2^{n^k})$.

Key relation:

$\mathrm{PSPACE} \subset \mathrm{EXPSPACE}$.

PSPACE problems can be solved in EXPSPACE, but not all EXPSPACE problems are in PSPACE.

2. Explain the PSPACE with example.

Solution:

PSPACE: Problems solvable with polynomial space.

Example: TQBF (True Quantified Boolean Formula)

- **Input**: ∃x ∀y (x ∨ ¬y)
- **Task**: Determine if the formula is true.
- **Explanation**: Solvable with polynomial space, but may require exponential time.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva – Questions

1. How does PSPACE relate to the classes P and NP?

Solution:

PSPACE contains both P and NP, as they use polynomial space.

2. What is the significance of PSPACE-complete problems?

Solution:

PSPACE-complete problems are the hardest problems in PSPACE, capturing complexity.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 24:

Reducibility

Concept Building

Reducibility

It is a fundamental concept in the Theory of Computation that deals with the relationship between computational problems. It involves transforming one problem into another in such a way that solving the new problem provides a solution to the original problem. This transformation helps in understanding the complexity of different problems and classifying them based on their computational difficulty.

Key Concepts of Reducibility:

Transformation of Problems:

Reducibility involves creating a procedure to convert instances of one problem (Problem A) into instances of another problem (Problem B). If such a procedure exists, Problem A is said to be reducible to Problem B.

Preserving Solutions:

The transformation must preserve the solution's correctness. This means that a solution to the transformed instance of Problem B should directly correspond to a solution to the original instance of Problem A.

Polynomial-Time Reduction:

A common form of reducibility is polynomial-time reduction, where the transformation procedure can be executed in polynomial time. This type of reduction is denoted as A≤pBA \leq_p BA≤p B, meaning Problem A is polynomial-time reducible to Problem B.

Classifying Problems:

Reducibility helps in classifying problems into complexity classes, such as P, NP, and NP-complete. By showing that a problem can be reduced to an NP-complete problem, we can infer that the problem is at least as hard as the NP-complete problem.

Proving NP-Completeness:

To prove a new problem is NP-complete, it must be shown that the problem is in NP and that a known NP-complete problem can be reduced to it in polynomial time. This demonstrates that the new problem is as difficult as any problem in NP.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Importance:

Understanding Complexity:

Reducibility allows us to understand the relative complexity of different problems. If a problem can be reduced to a known difficult problem, it suggests that the original problem is also difficult.

Algorithm Design:

By reducing complex problems to simpler ones, we can leverage existing efficient algorithms to solve the original problems. This approach guides the development of new algorithms and optimization techniques.

Classifying Computational Problems:

Reducibility helps in classifying problems into complexity classes like P, NP, and NP-complete. This classification is crucial for understanding which problems can be solved efficiently and which are inherently difficult.

Proving Intractability:

Demonstrating that a problem is NP-complete using reducibility shows that the problem is unlikely to have a polynomial-time solution, guiding researchers away from seeking efficient algorithms for such problems and towards approximation or heuristic approaches.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Pre-Tutorial (To be completed by student before attending tutorial session)

1. What is reducibility in the context of computational theory?

Solution:

In computational theory, *reducibility* is transforming one problem into another to determine solvability or complexity.

2. What does it mean if problem A is reducible to problem B?

Solution

If problem A is reducible to problem B, solving B provides a solution for A. This means B is at least as hard as A in terms of complexity.

3. Why is reducibility important in computational complexity?

Solution:

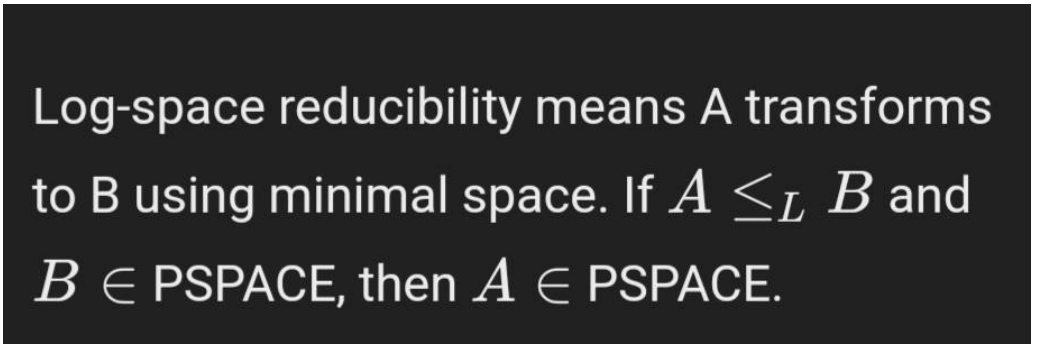
Reducibility helps classify problems by complexity, showing relationships between problems and identifying intractable or unsolvable problems.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

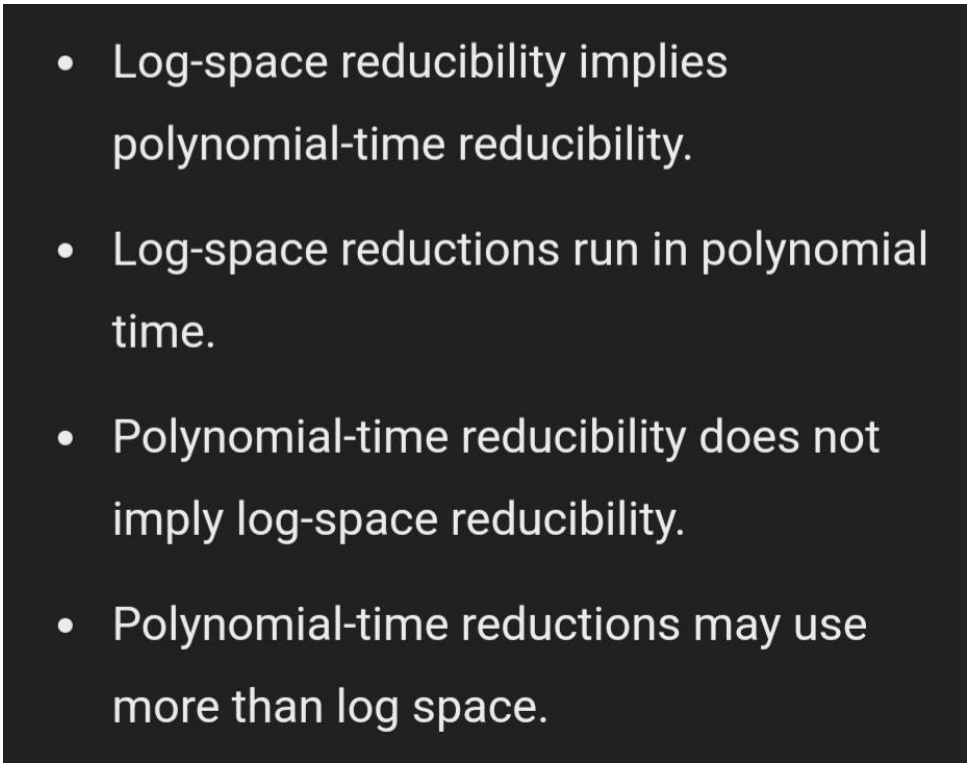
1. Explain the concept of log-space reducibility. If A ≤L B and B ∈ PSPACE, what can be concluded about A?

Solution



2. What is the relationship between log-space reducibility and polynomial-time reducibility? Does one imply the other?

Solution:



Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. Explain the concept of polynomial-time reducibility. If A ≤P B and B ∈ NP, what can be concluded about A?

Solution:

Polynomial-time reducibility means A transforms to B in polynomial time. If $A \leq_P B$ and $B \in$ NP, then $A \in$ NP.

4. Suppose A ≤L B and B ≤L C. What can be concluded about the relationship between A and C?

Solution:

If $A \leq_L B$ and $B \leq_L C$, then $A \leq_L C$. This means A is log-space reducible to C, as log-space reductions are transitive.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Post-Tutorial (To be carried out by student after attending tutorial session)

1. If A ≤P B and B ∈ EXPTIME, what can be concluded about the complexity of A?

Solution:

If $A \leq_P B$ and $B \in$ EXPTIME, then $A \in$ EXPTIME. This is because a polynomial-time reduction from A to B implies A can be solved within the same time complexity as B, which is EXPTIME.

2. Define polynomial-time reducibility (≤P) and explain its significance in computational complexity.

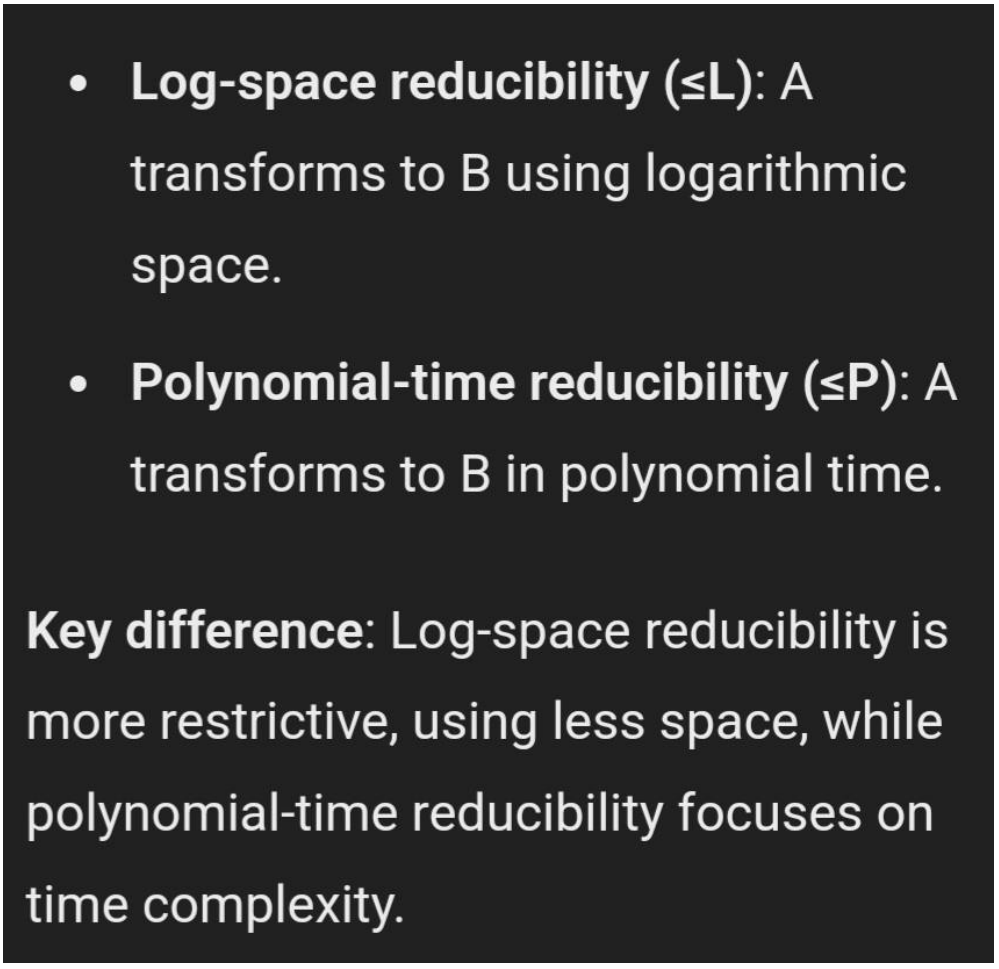
Solution

Polynomial-time reducibility (≤P): A transforms to B in polynomial time.
Significance: Helps classify problem difficulty and establish problem hardness.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

3. What is the difference between log-space reducibility (≤L) and polynomial-time reducibility (≤P)?

Solution:

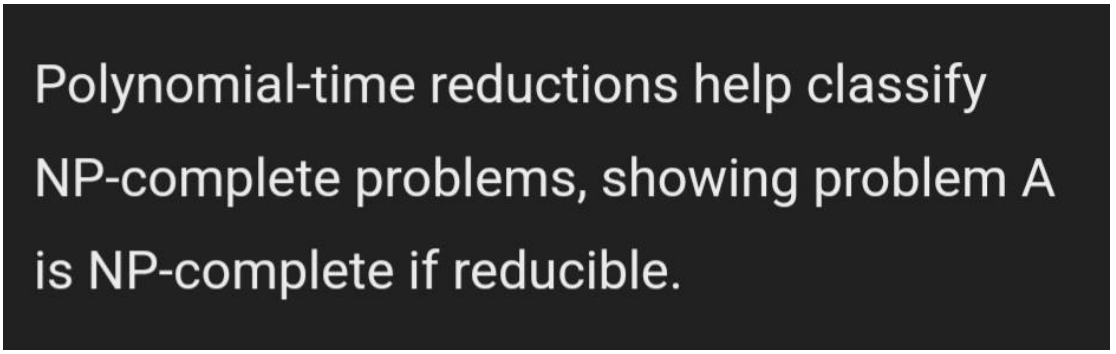


Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Viva – Questions

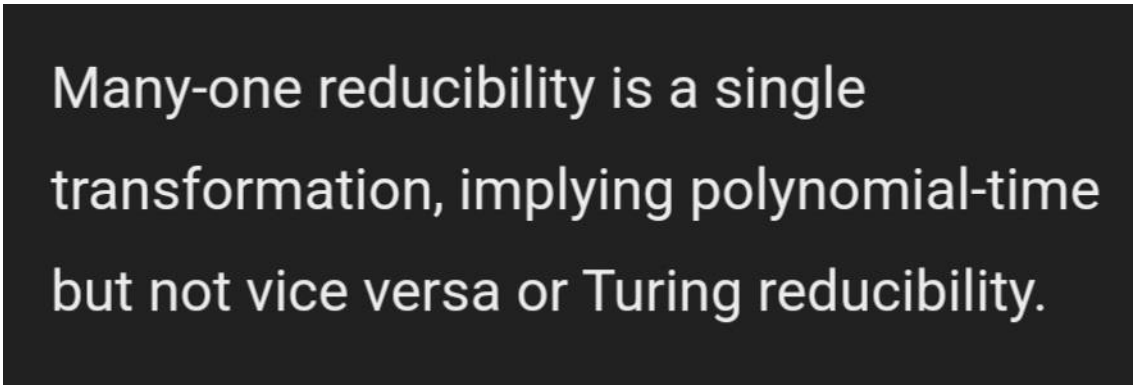
1. What is the significance of polynomial-time reductions in the context of NP-completeness?

Solution:



2. Explain the concept of many-one reducibility and its relationship to other types of reducibility.

Solution:



(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: