PUNE INSTITUTE OF COMPUTER TECHNOLOGY

27, DHANKAWADI, PUNE 411 043.

A.Y. 2023-24

DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

[6003] - 701

TE (Information Technology)

Theory of Computation

2019 pattern (Sem-I) (314441)

Time: 2½ hrs. Max. Marks - 70

End sem - Solution.

Q.1 a Eliminate useless symbols from the following grammar (6M) CO3

$S \rightarrow aA \mid bB$

$A \rightarrow aA \mid a$

$B \rightarrow bB$

→ (a) Elimination of Non-generating symbols

Generating symbols

as $A \rightarrow a$ $S \rightarrow aA$.

G.S. = { A, S }

(b) (a, b) $A \rightarrow a$ → (a, b, A) $S \rightarrow aA$ → (a, b, A, S)

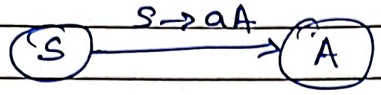
~~$S \Rightarrow aA$~~

B is Non-generating CO3

$S \rightarrow aA$

$A \rightarrow aA \mid a$

(b) ~~Non~~ Eliminate Non-reachable Symbol



All are reachable i.e. S & A.

Simplified grammar

$G = (V = \{S, A\}, T = \{a, b\}, P = \{S \to aA, A \to aA \mid a\}, S)$

Q.1 b.) Prove that CFL's are closed under Union, concatenation, Kleen's closure. (6M)

2 marks each

→ (1) Union — 2M

Let L_1 - be CFL generated by $G_1 = (V_1, T_1, P_1, S_1)$

Let L_2 - be CFL generated by $G_2 = (V_2, T_2, P_2, S_2)$

Cos. G be grammar generated by $G_1 \cup G_2$

$G_2 = (V, T, P, S)$

$G = (V = (V_1 \cup V_2) \cup S, T = (T_1 \cup T_2), P = (P_1 \cup P_2, \cup S \to S_1 | S_2), S)$.

S can ~~be~~ generated string of terminals either by selecting start symbol S_1 of G_1 or start symbol S_2 of G_2.

S can generate a string from L_1 or from L_2.

$\therefore L(G) = L_1 \cup L_2$

= Concatenation $L = L_1 \cap L_2$. (2M)

Kleen closure $L = L^*$ $S \to SS_1 / \epsilon$. (2M)

Q. 1 c What is an ambiguous grammar? Explain with a suitable example?

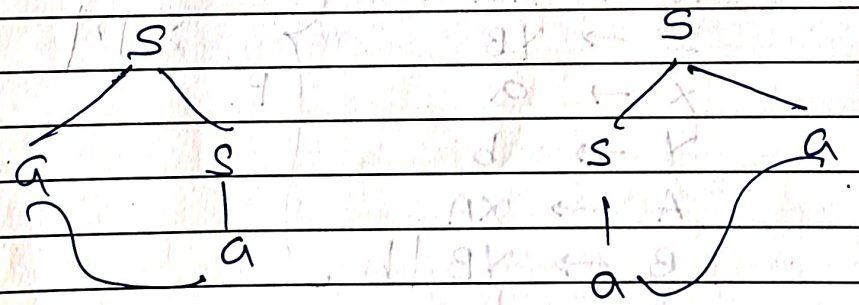
Ans Def:- " content free grammar with more than one distinct parse tree for same string is called as ambiguous grammar. " (3M)

$S \to aS \mid Sa \mid a$

(03)

$V = \{S\}$, $T = \{a\}$

"aa"



Two distinct parse trees for 'aa' so the grammar is ambiguous.

Q.2.

a Convert the following CFG to Chomsky Normal Form (CNF) (8M)

$S \rightarrow aAbB$

$A \rightarrow aA$

$B \rightarrow bB \mid b$

$X \rightarrow a \quad Y \rightarrow b$ } 1M

$S \rightarrow XAYB$
$W \rightarrow XA$
$Z \rightarrow YB$ } 2M

$\therefore S \rightarrow WZ$
$A \rightarrow XA$
$B \rightarrow YB \mid b$ } 2M

CNF:

$S \rightarrow WZ$
$W \rightarrow XA$
$Z \rightarrow YB$
$X \rightarrow a$
$Y \rightarrow b$
$A \rightarrow XA$
$B \rightarrow YB \mid b$ } P. 2M

1M $G = (V = \{S, W, Z, X, Y, A, B\}, T = \{a, b\}, P, S)$.

Q.2b) Construct NFA for the following left linear grammar (6M)

$S \rightarrow B1 \mid A0 \mid C0$

$B \rightarrow B1 \mid 1$

$A \rightarrow A1 \mid B1 \mid C0 \mid 0$

$C \rightarrow A0$

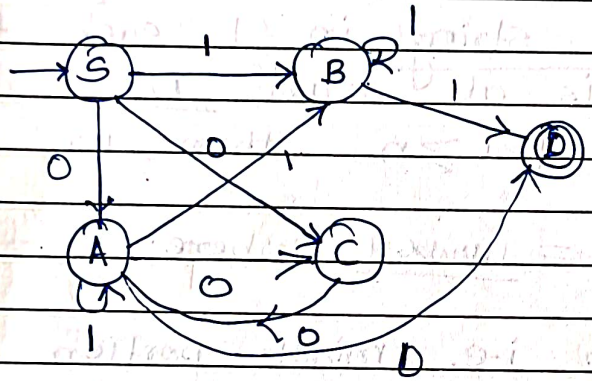
DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

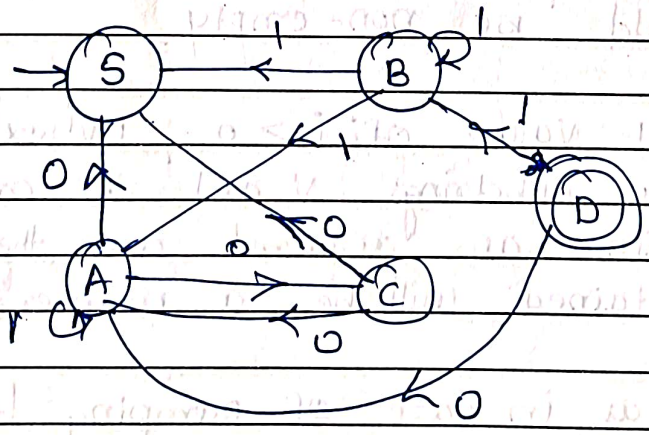
→ Step-I TG for give LLG



2M

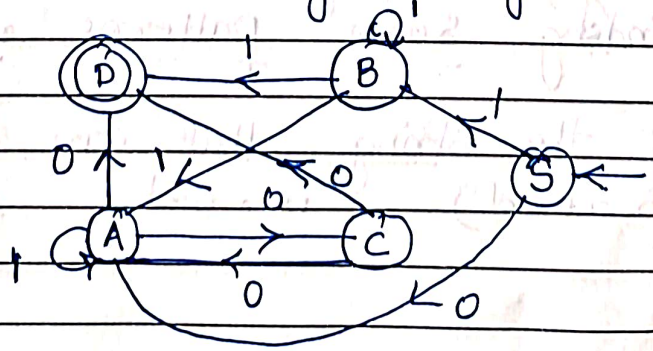
D is final state
S is initial state

Step-II Reverse Directions of all transitions



2M

Step-III: Interchange Starting and the final state



Resultant NFA

8M

Step 2.

Q.2 C) Write a note on Pumping lemma for CFL. (4M)

- Pumping lemma can be used to check whether or not a given language is a CFL.

Lemma Statement : Let L be a CFL. Then there exists a constant 'n' such that if z is any string in L, such that it's length is at least n.

2M

i.e. $|z| \geq n$ then we can write

$z = uvwxy$ where:

1) $|vwx| \leq n$ i.e. middle portion of the string is not too long.

2) $|vx| \geq 1$ i.e. $vx \neq \epsilon$ since v & x are the substrings that get pumped, it is required that at least one of them should be non-empty.

2M conditions.

3) For all values of $i, i \geq 0$, uv^iwx^iy is in L. i.e. the two substrings v and x can be pumped as many times as required and the resultant string obtained will be a member of L.

Just as in case of pumping lemma for regular lang, Pumping lemma for CFLs suggest finding some patterns near the middle of the string that can be pumped (or repeated) in code

Q-3

a Design a PDA for the lang (6M)

$L = \{a^n b^{2n} \mid n > 0\}$

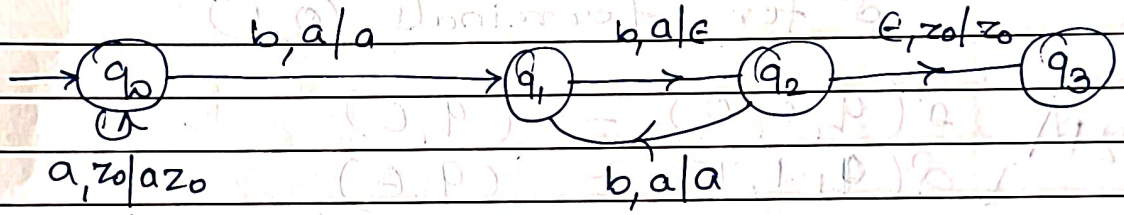
→ $L = \{abb, aabbbb, aaabbbbbb, ----\}$

Logic 2M

① Push all a's into stack. at q_0 State

② For 2 b's pop a 'a' from Stack at q_1 & q_2.

T.D. 2M



CO4

2M Transition.

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, a)$

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

$\delta(q_2, b, a) = (q_1, a)$

$\delta(q_2, \epsilon, z_0) = (q_3, z_0)$ → empty stack.

Q.3 b) Construct PDA equivalent to the following CFG. (6M)

$S \rightarrow 0BB \quad B \rightarrow 0S \mid 1S \mid 0$

→

PDA $M = (\{q\}, \{0,1\}, \{0,1,S\}, \delta, q, S, \phi)$

δ given by,

1M $\delta(q, \epsilon, S) = \{(q, 0BB)\}$

1M $\delta(q, \epsilon, B) = \{(q, 0S), (q, 1S), (q, 0)\}$

δ for terminals (0,1)

2M $\delta(q, 0, 0) = (q, \epsilon)$

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

eg: "000"

$\delta(q, 000, S) \xrightarrow{\delta(q, \epsilon, S) = (q, 0BB)} (q, 000, 0BB)$

$\xrightarrow{\delta(q,0,0) = (q, \epsilon)} (q, 00, BB)$

(2M) $\xrightarrow{\delta(q, \epsilon, B) = (q, 0S)} (q, 00, 0B)$

CO4 $\xrightarrow{\delta(q,0,0) = (q, \epsilon)} (q, 0, B)$

$\xrightarrow{\delta(q, \epsilon, B) = (q, 0)} (q, 0, 0)$

$\xrightarrow{\delta(q,0,0) = (q, \epsilon)} (q, \epsilon, \epsilon)$

Accepted

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

27, DHANKAWADI, PUNE 411 043.

DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

Q.3 CO4

c. Write a note on post Machine (PM) (5)

CO4

→ PM can accept the set of languages accepted by FSM, PDM as well as TM

Defn. 1M

- The major difference betn PDA & PM is that PM uses queue data structure as memory, while PDA uses stacks.

- PM more powerful than PDA, it's equivalent to TM

Elements of PM

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

2M

① A tape, which is bounded on one end and unbounded on the other.

② Input string initially written onto the tape using the special char. "#" to indicate the end. the rest of the tape is blank.

② A finite set of allowable tape symbols (Γ) including '#' and blank character ß

③ A finite input alphabet Σ, which is a subset of the set of allowable tape symbols, that is $\Sigma \subset \Gamma$.

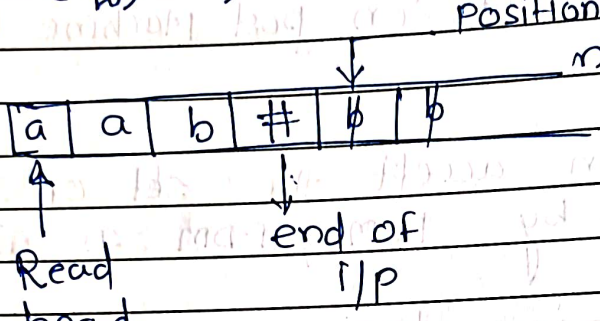
4. A start state. q_0

5. "accept" state and 'F'

6. Front symbol of the queue (tape)

7. 'δ'

$$\delta(q_0, a) = (q_1, b)$$



q_0 → current state

Thus, in a PM i/p written onto it's tape the machine reads the tape symbols & if necessary add symbols to the end of i/p string (rear).

PM can write symbols onto it's tape just as the TM does.

The major difference bet^n the TM and PM is that the head of the PM is allowed to move towards the right only.

It can not move towards left or remain the same position like TM.

Transition δ in PDA means

$$\delta(q_0, a) = (q_1, b)$$

(1) q_1 new / next state

(2) An insert with delete

(3) An insert without delete

(4) A delete without insert

(5) No delete no insert

(2) $\delta(q_0, a) = (q_1, b)$

↓ delete 'a' from front

↓ insert 'b' at rear

(3) An insert w/o delete

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

↓ No delete

↑ insert 'b'

(4) Delete w/o insert

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

↓ delete 'a' from front

↓ No. insert

(5) No delete no insert

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

↓ No del

↓ No insert.

Q.4 a Design a PDA which accepts only odd no. of b's over $\Sigma = \{a, b\}$. Simulate PDA for String "bbaba" (6M)

→ $L = \{b, ab, ba, aba, aab, baa, ababb, \dots\}$

Logic

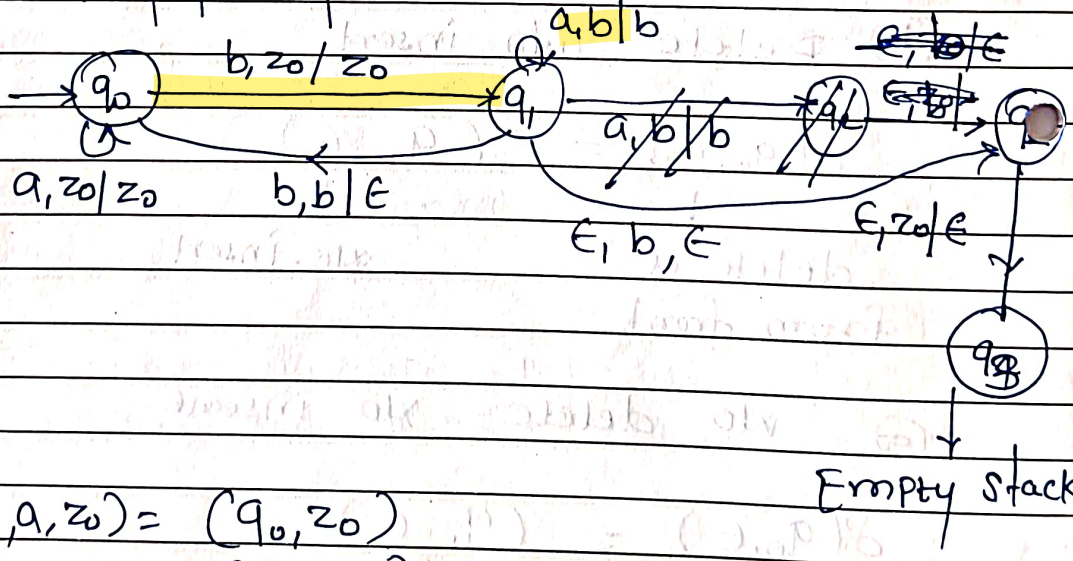
① Read 'a' with no operation

② Read even 'b' with pop operation from q_1 to q_0

③ Read odd 'b' with push operation from q_0 to q_1

④ 'a' at q_1 with no operation

⑤ When string is empty move from q_1 to q_2 with pop operation.



$\delta(q_0, a, z_0) = (q_0, z_0)$

$\delta(q_0, b, z_0) = (q_1, z_0)$

$\delta(q_1, a, b) = (q_1, b)$

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

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

$\delta(q_2, \epsilon, z_0) = (q_3, \epsilon)$

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

27, DHANKAWADI, PUNE 411 043.

DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

Q.4 CO4

b) Explain the acceptance by PDA (6M)

i) By final state

ii) By empty stack.

→ - PDA is pushdown Automata with Stack

- PDA can be deligned/accepted by 2 ways.

i) By final state:

Let PDA $M = (Q, \Sigma, T, \delta, q_0, z_0, F)$ then the language accepted by M through a final state is given by:

$$L(M) = \{ w \mid (q_0, w, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_1, \epsilon, \alpha) \}$$

Where, the state $q_1 \in F$

'α' the final contents of the stack are irrelevant as a string is accepted through a final state.

(2) By

(2) Acceptance By Empty Stack.

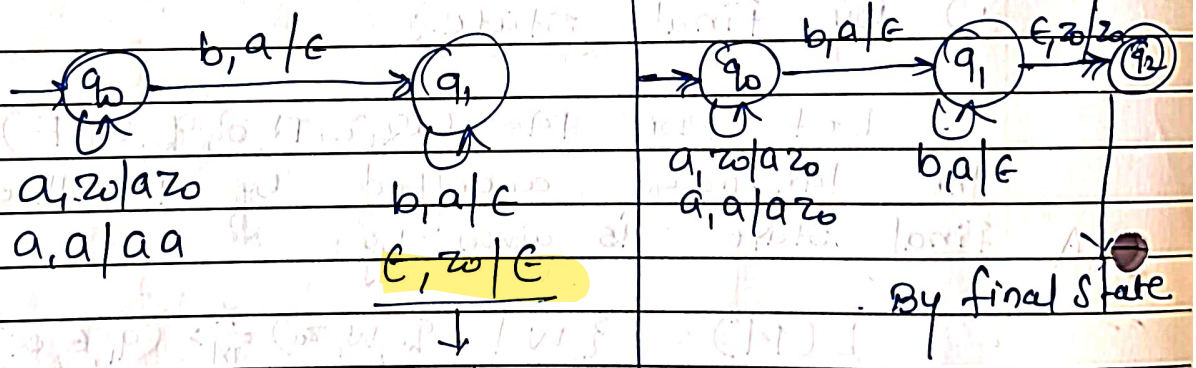
Let the PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi)$ then the language accepted through an empty stack is given by:

$$L(M) = \{ w \mid (q_0, w, z_0) \overset{*}{\underset{M}{\Longrightarrow}} (q_1, \epsilon, \epsilon) \}$$

Where q_1 is any state belonging to Q and the stack becomes empty on application of i/p string w.

(3M) including example

eg $L = \{ a^n b^n \mid n \geq 1 \}$



Q.4

c) Define pushdown Automata (3M)

"pushdown Automata is machine description of context free grammar/ CFL with secondary memory stack."

A PDA M is defined with 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

Q = The set of states
Σ = Input alphabet
Γ = stack symbols
δ = The transition f^n is a transition form $Q \times (\Sigma \cup \epsilon) \times \Gamma$ to $Q \times \Gamma^*$

q_0 = $q_0 \in Q$ is the initial state
F = $F \subseteq Q$ is the set of final states.
z_0 = An initial stack symbol

δ transition f^n

$Q \times (\Sigma \cup \epsilon) \times \Gamma$ to $Q \times \Gamma^*$

Where,

$Q \times (\Sigma \cup \epsilon) \times \Gamma$ implies that a transition is based on:

1) Current state
2) Next input (including ϵ)
3) Stacks symbol (topmost)

$Q \times \Gamma^*$ implies that:

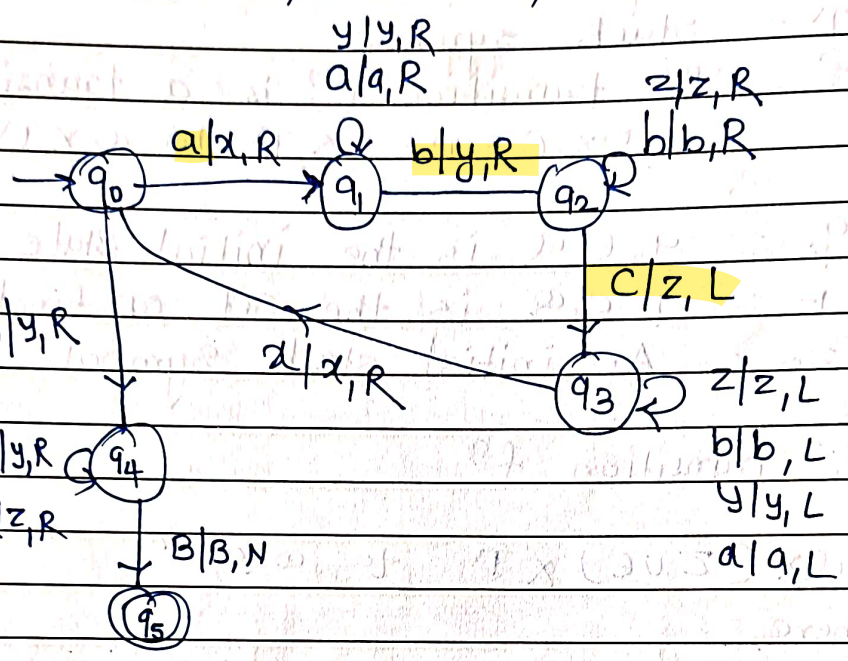
1. The next state could be any state belonging to Q.
2. It can perform push, pop or no-op on stack.

Q.5 a CO5

Design TM to accept language

$L = \{ a^n b^n c^n \mid n > 0 \}$ (TM)

→ $L = \{ abc, aabbcc, aaabbbccc, \ldots \}$



Transition Table (δ)

	a	b	c	x	y	z	B
→ q_0	(q_1, x, R)	–	–	–	(q_4, y, R)	–	–
q_1	(q_1, a, R)	(q_2, y, R)	–	–	(q_1, y, R)	–	–
q_2	–	(q_2, b, R)	(q_3, z, R)	–	–	(q_2, z, R)	–
q_3	(q_3, a, L)	(q_3, b, L)	–	(q_0, x, R)	(q_3, y, L)	(q_3, z, L)	–
q_4	–	–	–	–	(q_4, y, R)	(q_4, y, R)	(q_5, B, N)
* q_5	–	–	–	–	–	–	–

Components of TM

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

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

$\Gamma = \{ a, b, c, x, y, z, B \}$

Initial state = q_0

Blank symbol = B

$F = \{ q_5 \}$

PUNE INSTITUTE OF COMPUTER TECHNOLOGY
27, DHANKAWADI, PUNE 411 043.

DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

Q.5 b

CO5

Write a short note on Post correspondence problem. (PCP) (5M)

→ - Post correspondence problem (PCP) is undecidable problem.

- It is simpler than Halting Problem.

Def?

eg. In this problem we have N no. of Domino (tiles)

- The aim is to arrange tiles in such order that string made by Numerators is same as string made by denominators.

eg. with explanation

- i.e. let us assume we have 2 lists both containing n words, aim is to find out concatenation of these words in some sequence such that both lists yeild same result.

Let list A & B

A = [aa, bb, abb] & B = [aab, ba, b]

Now for seq. 1, 2, 1, 3 first list will yeid aabbaaabb

Second list will yeild same string "aabbaaabb"

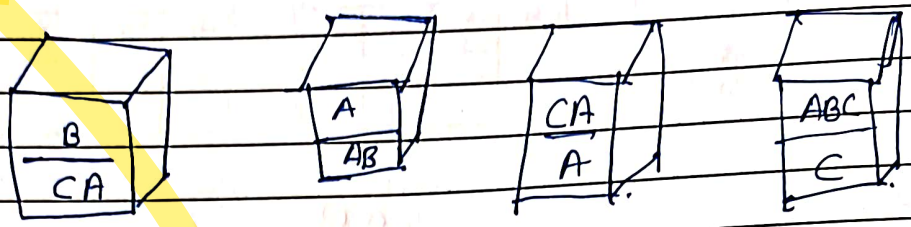
So the solution to this PCP becomes 1, 2, 1, 3

5M

3M

PCP can be represented in 2 ways.

1. Domino's Form:



② Table form

	Numerator	Denominator
1	B	CA
2	A	AB
3	CA	A
4	ABC	C

CO5

Q.5 C. Differentiate bet^{n} PDA and TM (6)

each 3M

	PDA	TM
①	Type of automata used Stack as secondary memory.	① TM uses input tape which is infinite
②	Less powerful than TM	② Most powerful than FA, PDA, PM.
③	NO read write head.	③ Have read write head.
④	PDA can access only the top of its stack.	④ TM can access any position on an infinite tape.

	PDA	TM
(5)	CFL can be recognized by PDA, i.e type 2 grammar,	(5) TM can recognized unrestricted lang i.e. type 0 grammar
(6)	PDA with 6 components	(6) TM with 6 components
	$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F/\phi)$	$M = (Q, \Sigma, T, \delta, q_0, B, F)$
(7)	'δ' transition fn in PDA	(7) 'δ' in TM
	$\delta(q_0, \Sigma, T_0) = (q, T^*)$	$\delta(q_0, T) = (q, T, L/R/N)$
	T → Stack top	where T → tape symbol
	T^* → stacks operation	L - Left Direction
	Push	R - Right →
	Pop	N - NO —
	No operation	
(8)	eg. Lang. accepted by PDA	(8) Lang. accepted b TM
	$L = \{a^n b^n \mid n>0\}$	$L = \{a^n b^n \mid n>0\}$
	$L = \{a^i b^j \mid i>j\}$	$L = \{a^n b^n c^n \mid n>0\}$

CO5

Q.6

a) Explain church Turing hypothesis (3)

Def → TM is defined as an abstract representation of a computing device such as hardware in computers.

Alan Turing proposed logical Computing machine (LCM) i.e Turing's expressions for TM. This was done to define algo property.

So, Church made a mechanical method named as 'M' for manipulation of strings by using logic and mathematics.

This method M must pass the following statements:

- No. of instructions in M must be finite.
- A o/p should be produced after performing finite no. of steps.
- It should ~~be~~ not be imaginary i.e. can be made in real life.
- It should not require any complex understanding.

explanation

Using these statements Church proposed a hypothesis called

Church's Turing hypothesis

"The assumption that the intuitive notion of computable functions can be identified with partial recursive f°."

i-e,

"Every computation that can be carried out in the real world can be effectively performed by a Turing machine."

Q.6 b. Design a Turing machine to add two unary nos. (7)

PUNE INSTITUTE OF COMPUTER TECHNOLOGY
27, DHANKAWADI, PUNE 411 043.

DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

Q.6.

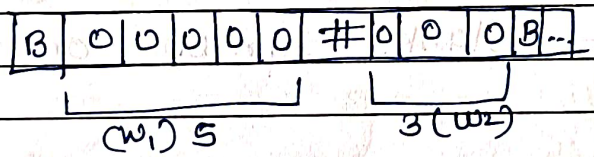
b. Let $\Sigma = \{0\}$

Addition can be performed through append operation.

eg. add 5 (w_1) & 3 (w_2)

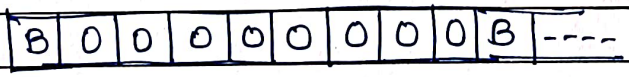
w is unary no.

1) Initially tape will be



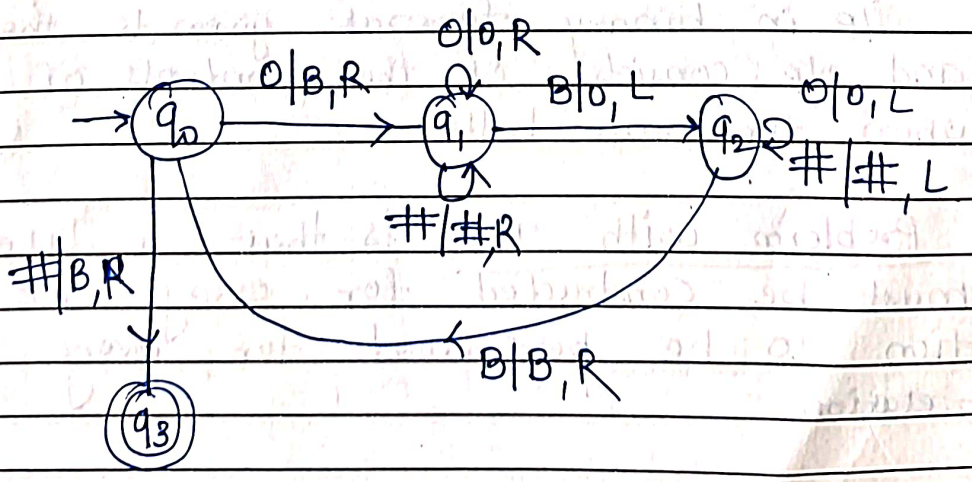
1M.

2) w_1 is appended to w_2



Every '0' from w_1 appended to w_2, '0' from w_1 is erased, w_2 contains 8 '0's addition of 5 & 3.

3M



Transition Table of 'δ'

	0	#	B	B
→ q_0	(q_1, B, R)	$(q_3, B, R$		–
q_1	$(q_1, 0, R)$	$(q_1, \#, R)$		$(q_2, 0, L)$
q_2	$(q_2, 0, L)$	$(q_2, \#, L)$		(q_0, B, R)
* q_3	–	–		–

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

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

$\Sigma = \{0\}$, $T = \{0, \#, B\}$

initial state = q_0

Final states = $F = \{q_3\}$

B - Blank symbol on the tape.

Q.6

C. Write a short note on Universal Turing Machine (UTM) (8 M)

→ Turing machine is the machine level equivalent to a digital computer.

- TM consists of an input and output i/p in binary format given to the tape, and o/p consists of the contents of the tape when the m/c halts.

- Problem with TM is that a different m/c must be constructed for every new computation to be performed for every i/p, o/p relation.

This is the reason the Universal Turing machine was introduced which along with i/p on the tape takes the description of a machine M.

The UTM can go on then to simulate M on the rest of the content of the input tape.

A UTM thus can simulate any other m/c.

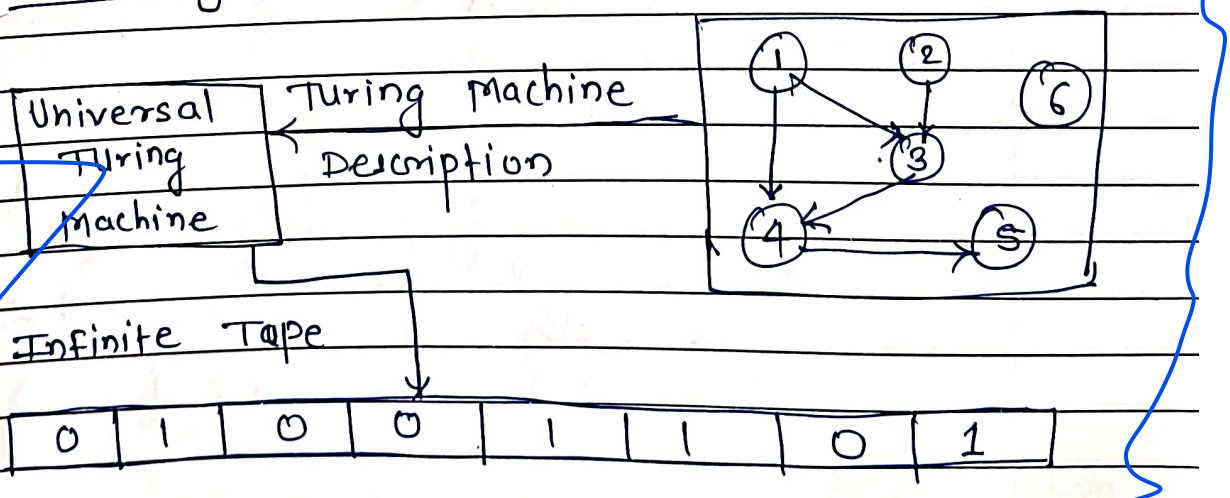
UTM would have 3 bits of info for the m/c it is simulating.

① A basic description of the m/c
② The content of machine tape
③ The internal state of the m/c.

The UTM would simulate the m/c by looking at the input on the tape and the state of the machine.

It would control the machine by changing it's state based on the input.

This leads to an idea of "computer running another computer"



CO6

Q.7

b Explain the Satisfiability problem with example (6M)

→

Satisfiability problem is used to determine if a given propositional formula is true for some assignment of truth values to the variables in the formula.

What is it? & DM

i.e. the problem of determining if the variables in a given Boolean formula can be assigned in such a way as to make the formula evaluate to true.

CNF – satisfiability is the satisfiability problem for CNF formulae.

Explain with ex. & AM

DNF – Satisfiability is the satisfiability problem related to DNF formulae.

A formula is said to be satisfiable if it is possible to find truth assignments to its variables that make the formula true.

For example, the formula $(a \wedge b)$ is satisfiable because we can find the values a=true and b=true, which makes $(a \wedge b)$ = true

A formula is said to be valid if all truth values of its variables make the formula true

A formula is said to be unsatisfiable if none of the truth assignments of its variables make the formula true.

A formula is said to be invalid if some such truth assignments to its variables make the formula false.

The worst-case complexity of a deterministic algorithm to find the satisfiability of a given formula of n variables is $O(2^n)$

The deterministic algorithm for the satisfiability problem requires exponential time.

It is easy to obtain a polynomial time non-deterministic algorithm that terminates succesfully if and only if a given propositional formula of n variables is satisfiable.

Q. 7 - CO6

c) What is polynomial time reduction? Explain it with a suitable example. (6M)

→ "A reduction is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem.

"Method for proving that problems are computationally unsolvable. It is called reducibility.

eg. Reducibility's come up often in everyday life.

eg - Want to find way around a new city.

- it would be easy if you had a map.

-Thus reduce the problem of finding your way around the city to the problem of obtaining a map of the city.

- Reducibility always involves 2 problems A & B

if A reduces to B, we can use a solution of B to solve A.

in our example

A — finding your way around the city.

B - obtaining a map.

Reducibility says nothing about solving A or B alone, but only about the solvability of A in the presence of a soln to B.

Eg2 Problem of measuring the area of a rectangle reduces to the problem of measuring its length and width.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

27, DHANKAWADI, PUNE 411 043.

DEPT. CLASS DIV ROLL NO. DATE

SUBJECT

Questions	1	2	3	4	5	6	7	8	Total
Marks obtained									

Examiner

Q. 8 CO6

a. Show that for 2 recursive languages L_1 & L_2, $L_1 \cup L_2$ is also recursive (4 M)

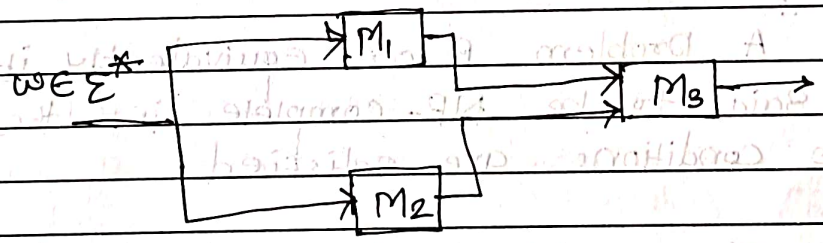
→ Let the TM M_1 decides L_1 & M_2 decides L_2

If a word $w \in L_1$ then M_1 returns 'Y' else it returns 'N'.

Similarly, if a word $w \in L_2$ then M_2 returns 'Y' else it returns 'N'.

2 M

Lets construct TM M_3,



TM for $L_1 \cup L_2$

2 M

- O/P of M_1 is written on the tape M_3
- O/P of M_2 is written on the tape of M_3.
- The machine M_3 returns "Y" as output, if at least one of the outputs of M_1 or of M_2 is 'Y'
- It should be clear that M_3 decides $L_1 \cup L_2$ As both L_1 & L_2 are turing decidable, after

a finite time both M_1 & M_2 will halt with answer "Y" or "N".

The machine M_3 is activated after M_1 and M_2 are halted.

The machine M_3 halts with answer 'Y' IF $w \in L_1$ or $w \in L_2$ else M_3 halts with output "N".

Thus, $L_1 \cup L_2$ is turing decidable or $L_1 \cup L_2$ is recursive.

Q. 8

b. What do you mean by NP-complete problems? List the problems in this class and explain any one problem in detail. (7)

→

An NP-complete problem :-

"A problem P or equivalently its language L_1 is said to be NP-complete if the following two conditions are satisfied.

2M

1) The problem L_2 is in the class NP.
2) For any problem L_2 in NP, there is a polynomial-time reduction of L_1 to L_2

More explicitly, a problem is NP-complete if it is in NP and for which no polynomial time Deterministic TM solution is known so far.

The most interesting aspect of NP-complete problems is that for each of these problems neither, so far it has been possiblem to design a deterministic p-time TM solving the problem nor it has been possible to show that deterministic

P-time TM should can not exist.

List of NP- complete problems

1. Satisfiability problem (SAT)
2. Travelling salesman problem (TSP)
3. Hamilton circuit problem (HCP)
4. The vertex cover problem (VCP)
5. clique problem
6. subgraph isomorphism problem
7. Exact cover problem

2M

Explain one of the above problem.

3M

CO6

Q-8 c. What do you mean by mapping Reducibility? Explain it with an example. (6M)

- What Reducibility - 3M.
- Example - 3M.