B.Tech.

FOURTH SEMESTER EXAMINATION, 2007-08

THEORY OF AUTOMATA & FORMAL LANGUAGES

(TCS-405)

Time : 3 Hours] *[Total Marks : 100*

Note : *(1) Attempt* ***all*** *questions.*
(2) All questions carry ***equal*** *marks.*
(3) Be precise in your answer.

Q. 1. Attempt any four parts of the following : $5 \times 4 = 20$

(a) Prove or disprove that

$(L_1 \cup L_2)^* = L_1^* \cup L_2^*$

where L_1 and L_2 are languages.

Ans. $(L_1 \cup L_2)^* = L_1^* \cup L_2^*$ is not true. We can disprove it by taking a counter example : Let regular expression 01 associated with L_1 and 10 is associated with L_2.

As we know :

$$L(r_1 + r_2) = L_1(r_1) \cup L_2(r_2)$$

but by given relation $[(L_1 \cup L_2)^* = L_1^* \cup L_2^*]$

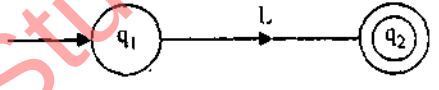
$$(01 + 10)^* = 01^* + 10^*$$

but above relation is not true because $(01 + 10)^*$ will give 0110 but $10^* + 10^*$ will never give 0110, hence above relation is not true.

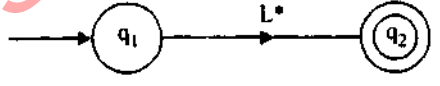
Q. 1. (b) Given a transition diagram for the language L. Show how to build a transition diagram for the language L^*.

Ans. Let transition diagram of L is as follows :

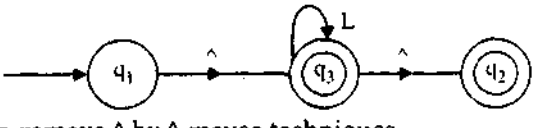
Where q_1 is initial state and q_2 is the final state. Now we can make a transition diagram for L^*



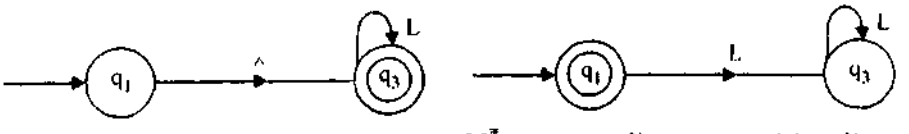
Step 1:



Step 2 :



Now, we can remove ^ by ^-moves techniques.



This is the final transition diagram of L^* corresponding to transition diagram L.

Q. 1. (c) Define nondeterministic finite automata ? How does it differ from deterministic finite automata ?

Ans. Non-deterministic finite Automata : A Non-deterministic finite Automata is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where ;

(i) Q is a finite non empty set of states ;

(ii) Σ is a finite non empty set of inputes;

(iii) δ is a transaction function mapping from

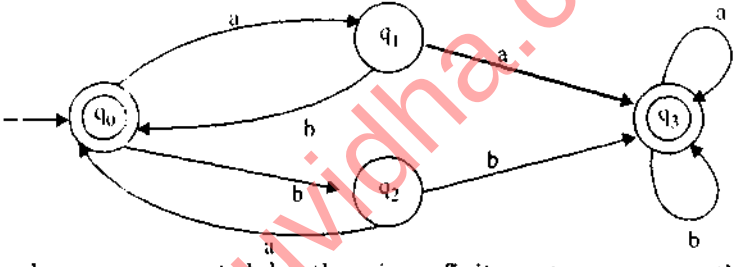
$Q \times \Sigma$ into 2^Q which is the power power set of Q the set of all subset of Q ;

(iv) $q_0 \in Q$ is the initial state and

(v) $F \in Q$ is the final states.

The difference between the deterministic and non deterministic automata is only in δ. For deterministic automata (DFA), the outcome is a state, i.e. an element of Q; for non deterministic automata the outcome is a subset of Q.

Q. 1. (d) Describe in words the language accepted by the following finite automata :



Ans. The language accepted by the given finite automaton are the strings given by $(ab + ba)^*$. As any of such strings is a string of ab's and ba's. We get an equal number of a and b. If a prefix x of a sentence accepted by the finite automata has an even number of symbols, then it should have an equal number of an a and b. If the prefix x has an odd number of symbols, then we can write x as ya or yb. As y has even number of symbols, y has an equal number of a's and b's. Thus x has one more a than b or vice-versa.

Q. 1. (e) Suppose that the language $L \subseteq \{a, b\}^*$ is defined as follows :

Rule 1 : $a \in L$

Rule 2 : any for $x \in L$, ax is in L.

Rule 3 : for any $x \in L$, xb is in L.

Rule 4 : No other strings are in L.

Describe the language L in terms of set.

Ans. Language L = set of all strings (expects $\wedge$) started with a and ending with b.

Q. 1. (f) Explain the Chomsky classification of languages.

Ans. Chomsky classified the grammars into four types in terms of production (types 0-3).

Type - 0 → A type-0 grammar is any phrase structure grammar without any restriction. In a production of the form $\phi A \psi \rightarrow \phi \propto \psi$ where A is a variable, ϕ is called the left context ψ the right context, and $\phi \propto \psi$ the replacement string.

Type-1 → A grammar is called type-1 or context sensitive context department if all its production are type-1 production. A production if $\alpha \neq \wedge$. In type-1 productions, erasing of A is not permitted.

Type-2 → A type-2 production in a production of the form $A - \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup \Sigma)^*$. In other words, the LHS has no left context or right context.

Type-3 → A grammar is called a type-3 or regular grammar if all its production or type-3 production. A production $S \to \wedge$ is allowed in type-3 grammar, but in care S does not on the right -hand side of any production

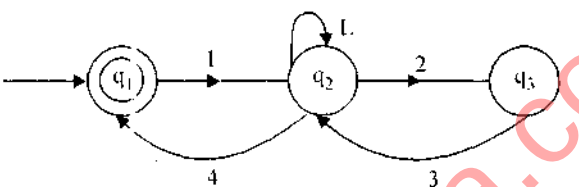
Q. 2. Attempt any two parts of the following : **10 × 2 = 20**

(a) (i) Design a DFA which accepts the set of strings over alphabet $\Sigma = \{1, 2, 3, 4\}$ such that string when interpreted as decimal numbers, sum of their digits are divisible by 5.

Ans. $\Sigma = \{1, 2, 3, 4\}$

Let No. of states $Q = q_1, q_2, q_3$

Transition diagram :



Q. 2. (a) (ii) Simplify the following regular expressions

[A] $r_1 (r_1^* r_1 + r_1^*) + r_1^*$

[B] $(r_1 + r_2 + r_1 r_2 + r_2 r_1)^*$

Where r_1 and r_2 are regular expressions

Ans. [A] r_1^* It can be seen by transition diagram.

[B] $(r_1 + r_2)^*$. It can be seen by transition diagram.

[A] $r_1 (r_1^* r_1 + r_1^*) + r_1^*$ ...(i)

$r_1^* r_1 + r_1^*$ means set of strings either formed by r_1^* r_1 or by r_1^*

$r_1^* r_1 = r_1, r_1 r_1, r_1 r_1 r_1 \ldots\ldots$

$r_1^* = \wedge, r_1, r_1 r_1 \ldots\ldots$

So total string formed by r_1^* $r_1 + r_1$ is same as total strings formed by

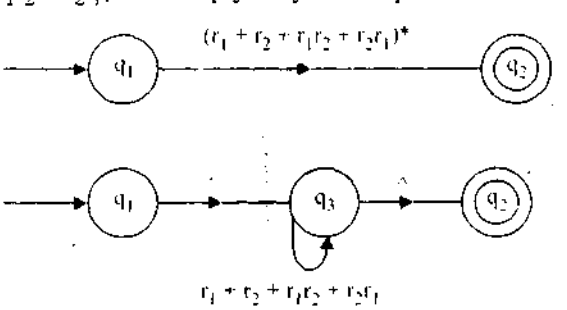
$r_1^* = \wedge, r_1, r_1 r_1 \ldots\ldots$ Hence, $\boxed{r_1^* r_1 + r_1^* = r_1^*}$ ...(ii)

Now from (i) equation : $r_1 (r_1^* r_1 + r_1^*) + r_1^* = r_1 r_1^* + r_1$ –by (ii)

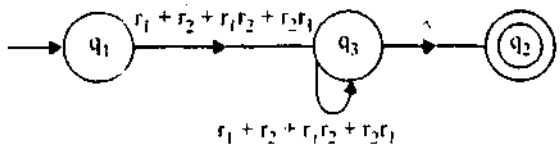
$= r_1^*$ -by – (ii)

Hence $\boxed{r_1 (r_1^* r_1 + r_1^*) + r_1^* = r_1^*}$

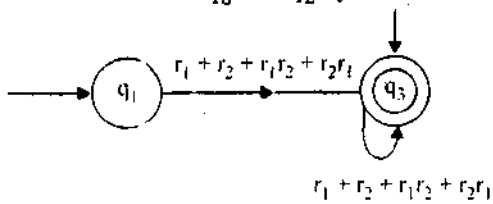
[B] $(r_1 + r_2 + r_1 r_2 + r_2 r_1)^*$ we simply it by the help of transition diagram



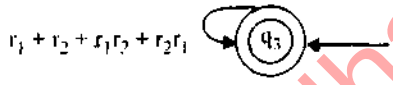
(i) Remove ^ between q_1 & q_3 by ^-move technique



(ii) Now remove ^ between q_3 and q_2 by ^ move technique.



q_3 & q_1 both are initial state. So final transition diagram is.



Hence simplified regular expression $= (r_1 + r_2 + r_1r_2 + r_2r_1)^*$

We can say the given expression in question is all ready simplified.

Q. 2. (b) (i) Write regular expression corresponding to the following languages in {0, 1} *.

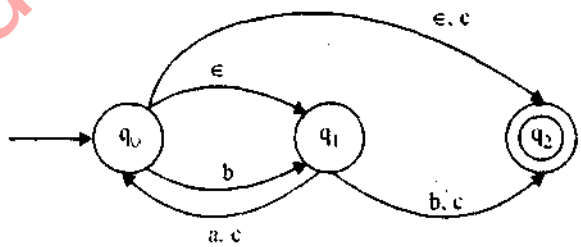
(a) The language of all strings in which every 0 is followed immediately by 11.

(b) The language of all strings that has at most one pair of consecutive 1's.

Ans. (a) $1^*(011)^*1^*$

(b) $(0 + 1 + 10 + 01)^*$

Q. 2. (ii) Convert following NFA to equivalent DFA and hence minimize the number of states in the DFA.



Ans. State table of NFA :

State/Σ	^	*a*	*b*	*c*
q_0	q_1, q_2	–	q_1	q_2
q_1	–	q_0	q_2	q_0, q_2
q_2	–	–	–	–

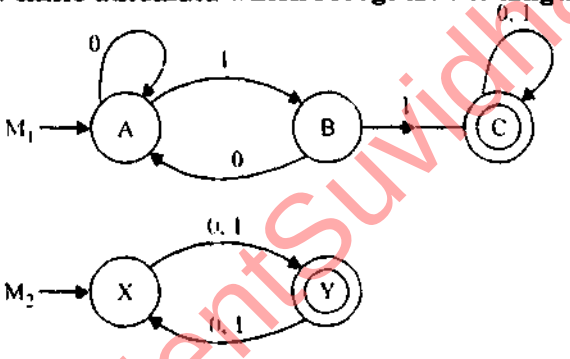
For DFA states are subset of $\{q_0, q_1, q_2\}$, Now, δ is defined for *DFA* as follows :

State/Σ	^	a	b	c
$[q_0]$	$[q_1, q_2]$	–	$[q_1]$	$[q_2]$
$[q_1]$	–	$[q_0]$	$[q_2]$	$[q_0, q_2]$
$[q_2]$	–	–	–	–
$[q_1, q_2]$	–	$[q_0]$	$[q_2]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_1, q_2]$	–	$[q_1]$	$[q_2]$

We can reduce the number of states in the state table when two rows are identical in the successes table. But it is not possible in this table. Hence we cannot minimize the number of states as shown above.

Q. 2. (c) (i) Let M_1 and M_2 be the FA recognizing the languages L_1 and L_2 respectively.

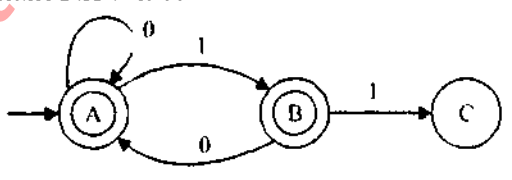
Draw finite automata which recognize the language $L_2 - L_1$.



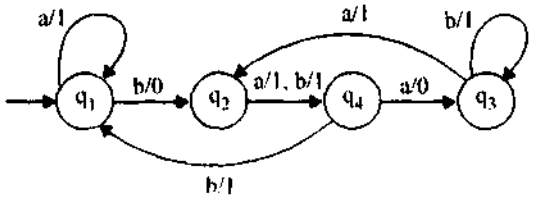
Ans. The finite automata which in recognize the language $L_2 - L_1$ will be design by FA M_1 with given rule.

Rule — Make non final state of M_1 and vice-versa

So, our resultant F.A will be



(ii) Transform the following Mealy machine into equivalent Moore machine.



Ans. Transition table of given Mealy machine

	Present state	Next state			
		Input a		Input b	
		State	Output	State	Output
→	q_1	q_1	1	q_2	0
	q_2	q_4	1	q_4	1
	q_3	q_2	1	q_3	1
	q_4	q_3	0	q_1	1

Now we can split q_i into several states. The number of such states being equal to the number of different outputs associated with q_i.

Now new table is constructed as given below

	Present state	Next state			
		Input a		Input b	
		State	Output	State	Output
→	q_1	q_1	1	q_{20}	0
	q_{20}	q_4	1	q_4	1
	q_{21}	q_4	1	q_4	1
	q_{30}	q_{21}	1	q_{31}	1
	q_{31}	q_{21}	1	q_{31}	1
	q_4	q_{30}	0	q_1	1

The pairs of states and output in the next state column can be rearranged as —

	Present state	Next state		Output
		a	b	
→	q_1	q_1	q_{20}	1
	q_{20}	q_4	q_4	0
	q_{21}	q_4	q_4	1
	q_{30}	q_{21}	q_{31}	0
	q_{31}	q_{21}	q_{31}	1
	q_4	q_{30}	q_1	1

Here we observed that the initial state q_1 is associated with output 1. This means that with output $\wedge$ we get an output 1, if the machine starts at state q_1. Thus Moore machine accepts a zero–length sequence which is not accepted by the Mealy machine. To over–come this situation, either we must neglect the response of a Moore machine to input $\wedge$ or we add a new starting state q_0, whose state transition identical with q_1 but whose output is 0

Present state	Next state		Output
	a	b	
→ q_0	q_1	q_{20}	0
q_1	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_{30}	q_1	1

Q. 3. Attempt any two parts of the following : **10 × 2 = 20**

(a) (i) State pumping lemma for regular expression.

(ii) Prove or disprove that the language L given by

$L = \{a^m b^n \mid m \neq n$, m and n are positive integers $\}$ is regular.

Ans. 3a (i) Pumping Lemma : Let $M = (Q, \Sigma, S, q_0, F)$ be a finite automata with n states. Let L be the regular set accepted by M. Let $w \in L$ and $|w| \geq m$. If $m \geq n$ then there exist x, y, z such that $w = xyz$ $y \neq \wedge$ and $xy^i z \in L$ for each $i \geq 0$.

Ans. 3a (ii) $L = \{a^m b^n\}$ $m \neq n$, m and n are positive integer $\}$ is regular. We can prove it by contradiction.

Let $m = n$

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

$\{\delta(q_0, a^n) \mid n \geq 0\}$ in a subset of Q and hence finite. So $s(q_0, a^n) = \delta(q_0, a^m)$

for same m and n ; $m \neq n$. So

$\delta(q_0, a^m b^n) = \delta(\delta(q_0, a^m), b^n) = \delta(\delta(q_0, a^n), b^n) = \delta(q_0, a^n b^n)$

As $a^n b^n \in L$, $\delta(q_0, a^n b^n)$ in a final state and so is $\delta(q_0, a^m b^n)$. This means $a^m b^n \in L$ with $m \neq n$. It mean given language is regular if and only if $m \neq n$.

Q. 3. (b) (i) Define context free grammar. Find a context free grammer for the following language :

$L = \{a^i b^j c^k \mid j \geq i + k$; $i, j, k\}$ are nonnegative integers.

Ans. A grammar is called context free grammar if it contains only type-2 production. A type-2 production in a production of the form $A \to \alpha$ where $A \in V_N$ and $\alpha_E (V_N \cup \Sigma)^*$

The CFG for the given language L is

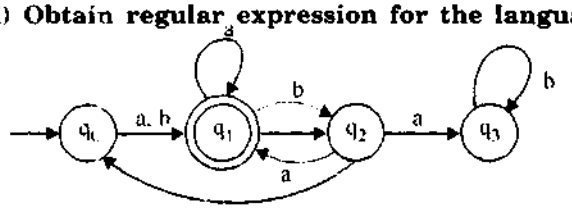
$G = (V_N, \Sigma, P, S)$ where P is

$S \to AB$

$A \to a\ A\ b \mid \wedge$

$B \to b\ B\ c \mid \wedge$

Q. 3. (b) (ii) Obtain regular expression for the language accepted by following automata :



Ans. The given question is wrong because transition from q_3 to q_0 is not labelled by any element.

Q. 3. (c) (i) For the given CFG, find an equivalent CFG with no useless variables

$S \rightarrow AB \mid AC$

$A \rightarrow a\ A\ b \mid b\ A\ a \mid a$

$B \rightarrow bbA \mid aaB \mid AB$

$C \rightarrow abC\ a \mid a\ D\ b$

$D \rightarrow bD \mid a\ C$

Ans. Given production of CFG (V_N, Σ, P, S) where production P are as follows :

$S \rightarrow AB \mid AC$

$A \rightarrow a\ A\ b \mid b\ A\ a \mid a$

$B \rightarrow bbA \mid aaB \mid AB$

$C \rightarrow abC\ a \mid a\ D\ b$

$D \rightarrow bD \mid a\ C$

Construction of P'- Now we can change the production of A as : $\boxed{A \rightarrow aab / baa / a}$

Now, we can change the production of B as :

$B \rightarrow bba / aabb\ A / aB$

$B \rightarrow bba / aabba / a\ bb\ A.$

$\boxed{B \rightarrow bba / aabba / a\ bb\ aab / abbbaa / aa}$

Now, we can make any change in variable C and D and it is also not terminated so it will not include in $V_{N'}$

Hence our final CFG is as follows :

$P' \rightarrow$

$S \rightarrow AB / AC$

$A \rightarrow aab / baa / a$

$B \rightarrow bba / aabba / abbaab / abbbaa / aa.$

Q. 3. (c) (ii) Explain Chomsky normal form and Greibach normal form. Convert the following CFG to equivalent Griebach normal forms.

$S \rightarrow AA$

$A \rightarrow SS$

$S \rightarrow a$

$A \rightarrow b$

Ans. The given grammar is in CNF. S and A are renamed as A_1 and A_2 respectively. So the productions are $A_1 \rightarrow A_1A_2 / a$ and $A_2 \rightarrow A_1A_1 / b$. As the given grammar has no null production and is in CNF we need not carry out step 1. So we proceed to step 2.

Step-2 : (i) A_1-Production are in the required form.

They are $A_1 \rightarrow A_2A_2 / a$

(ii) $A_2 \rightarrow b$ is in the required form. Apply

Lemma 6.1 to $A_2 \rightarrow A_1A_1$.

The resulting production are $A_2 \rightarrow A_2A_2A_1, A_2 \rightarrow aA_1$.

Thus, the A_2-production are :

$A_2 \rightarrow A_2A_2A_1, \quad A_2 \rightarrow aA_1, \quad A_2 \rightarrow b.$

Step-3 : we have to apply Lemma 6.2 to A_2 production as

we have $A_2 \rightarrow A_2A_2A_1$. Let z_2 be the new variable. The resulting production are :

$A_2 \to a\,A_1 \quad , \qquad A_2 \to b$

$A_2 \to aA_1, A_2 \quad , \quad A_2 \to bz_2$

$z_2 \to A_2A_1 \quad , \quad z_2 \to A_2A_1z_2$

Step-4 : (i) The A_2-production are $A_2 \to aA_1 | b | aA_1z_2 | bz_2$.

(ii) Among the A_1-productions we retain $A_1 \to a$ and eliminate $A_1 \to A_2A_2$ using Lemma a for GNF. The resulting production is

$A_1 \to a | aA_1A_2 | bA_2 | a\,A_1z_2A_2 | b\,z_2A_2$

Step-5 : The z_2-productions to be modified are $z_2 \to A_2\,A_1$, $z_2 \to A_2A_1z_2$. we apply Lemma 6.1 and get :

$z_2 \to a\,A_1A_1 / bA_1 / a\,A_1z_2A_1 | bz_2A_1$

$z_2 \to a\,A_1\,A_1z_2 | b\,A_1z_2 | a\,A_1z_2A_1z_2 | bz_2A_1z_2$

Hence, the equivalent grammar is :

$G = (\{A_1, A_2, z_2\}, \{a, b\}, P_1, A_1)$

where P_1 consists of

$A_1 \to a | aA_1A_2 | b\,A_2 | a\,A_1z_2A_1 | bz_2A_2$

$A_2 \to aA_1 | b | a\,A_1z_2 | bz_2$

$z_2 \to a\,A_1A_1 | bA_1 | a\,A_1z_2A_1 | bz_2A_1$

$z_2 \to a\,A_1A_1z_2 | bA_1z_2 | aA_1z_2A_1z_2 | bz_2A_1z_2$.

Q. 4. Attempt any two parts of the following : 10 × 2 = 20

(a) Define push down automata. Design a PDA for the following language :

$L = \{a^i\,b^j\,c^k \mid i = j$ or $j = k\}$.

Ans. A push down automata defined as :

(i) a finite nonempty set of states denote by Q,

(ii) a finite nonempty set of input symbols denoted by Σ,

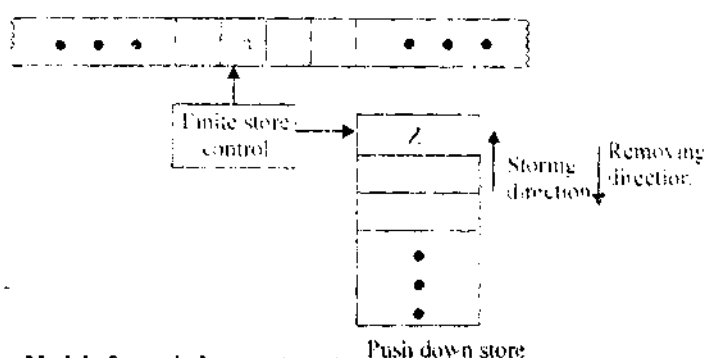
(iii) a finite non empty set of push down symbols denotes by Γ,

(iv) a special push down symbol called the initial symbol on the push down store denoted by z_0,

(v) a special state called the initial state denoted by q_0,

(vi) a set of final states, a subset of Q denoted by F,

(vii) a transition function δ from $Q \times (\Sigma \cup \{\wedge\}) \times \Gamma$ to the set of finite subset of $Q \times \Gamma^*$



Model of a push down automata

$L = \{a^i\,b^j\,c^k \mid i = j$ or $j = k\}$

As in given question either $i = j$ or $j = k$

Let $i = j$ means No. of a and b in L are equal.

Now, we can design a PDA. A which is defined as follows

$A = (\{q_0, q_1\}, \{a, b\}, \{a, z_0\}, \delta, q_0, z_0, \phi)$ δ is defined as follows :

$R_1 : \delta(q_0, \wedge, z_0) = \{(q_0, z_0)\}$

$R_2 : \delta(q_0, a, z_0) = \{(q_0, az_0)\}$

$R_3 : \delta(q_0, a, a) = \{(q_0, aa)\}$

$R_4 : \delta(q_0, b, a) = \{(q_1, \wedge)\}$

$R_5 : \delta(q_1, b, b) = \{(q_1, \wedge)\}$

$R_6 : \delta(q_1, c, z_0) = \{(q_1, z_0)\}$

$R_7 : \delta(q_1, \wedge, z_0) = \{(q_1, \wedge)\}$

Q. 4. (b) Consider the given PDA :

PDA $M = (\{q_0\}, \{0, 1\}, \{a, b, z_0\}, \delta, q_0, z_0, \phi)$

where δ **is defined as follows**

$\delta(q_0, 0, z_0) = \{(q_0, a z_0)\}$

$\delta(q_0, 1, z_0) = \{(q_0, bz_0)\}$

$\delta(q_0, 0, a) = \{q_0, aa\}$

$\delta(q_0, 1, b) = \{(q_0, bb)\}$

$\delta(q_0, 0, b) = \{(q_0, \in)\}$

$\delta(q_0, 1, a) = \{(q_0, \in)\}$

$\delta(q_0, \in, z_0) = \{(q_0, \in)\}$

Convert the given PDA M to the corresponding CFG.

Ans. PDA $M = (\{q_0\}, \{0, 1\}, \{a, b, z_0\}, \delta, q_0, z_0, \phi)$

Let $G = (V_N, \{0, 1, P, S\}$

where V_N consist of S_1 $[q_0, a, q_0], [q_0, b, q_0], [q_0, z_0, q_0]$

The production are

$P_1 : S \to [q_0, z_0, q_0]$

$\delta(q_0, 0, z_0) = \{(q_0, az_0)\}$ Yields

$P_2 : [q_0, z_0, q_0] \to 0\,[q_0, a, q_0]\,[q_0, a, q_0]$

$P_3 : [q_0, z_0, q_0] \to 0\,[q_0, z_0, q_0]\,[q_0, z_0, q_0]$

$\delta(q_0, 1, z_0) = \{(q_0, bz_0)\}$ Yields

$P_4 : [q_0, z_0, q_0] \to 1\ [q_0, b, q_0]$

$P_5 : [q_0, z_0, q_0] \to 1\,[q_0, z_0, q_0]$

$\delta(q_0, 0, a) = \{(q_0, a\ a)\}$ yields

$P_6 : [q_0, a, q_0] \to 0\,[q_0, a, q_0]$

$\delta(q_0, 1, b) = \{(q_0, b\ b)\}$ yields

$P_7 : [q_0, b, q_0] \to 1\ [q_0, b, q_0]$

$\delta(q_0, 0, b) = \{(q_0, \in)\}$ Yields

$P_8 : [q_0, b, q_0] \to 0$

$\delta(q_0, 1, a) = \{(q_0, \in)\}$ Yields

$P_9 : [q_0, a, q_0] \to 1$

$\delta(q_0, \in, z_0) = \{(q_0, \in)\}$ Yields

$P_{10} : [q_0, z_0, q_0] \to \wedge$

$P_1 - P_{10}$ gives the production in P of CFG.

Q. 4. (c) (i) Context free languages are closed under intersection. Prove the statement or give a counter example.

Ans. We can show it by taking a counter example : Let L_1 in a CFL and it is generated by grammar with productions.

$S \to ABC, A \to aAb \mid \wedge, B \to b\,B \mid b, C \to cC \mid \wedge$

and L_2 in a CFL and it is generated by the grammar with production.

$S \to AC, A \to aAc \mid B, B \to bB \mid \wedge, C \to cC \mid c$

Now for $L_1 \wedge L_2$ we cannot write any production which accept language generated by $L_1 \cap L_2$, So $L_1 \cap L_2$ is not CFL.

Q. 4. (c) (ii) Given a context free grammer G. Suggest an algorithm which can be used to decide whether language genrated by grammer G is empty or not.

Ans. Let $L = L(G)$ Language generated by the grammar $G = (V_N, \Sigma, P, S)$. To decide whether the language generated by grammar G is empty or not we can construct a PDA accepting L by empty store.

$A = \{(q_1), \Sigma, V_N \cup \Sigma, \delta, q, S, \phi\}$

where δ is defined as :

$R_1 : \delta(q, \wedge, A) = \{(q, \alpha) \mid A \to \alpha, \text{ as in } P\}$

$R_2 : \delta(q, a, a) = \{(q, 1)\}$ for every a in Σ.

Suggestions to write an algorithm to decide whether the language generated by grammar G is empty or not :

The push down symbol in PDA are variables and terminals. If the PDA reads a variable A on the top of PDS, it makes a $\wedge$ more by planing the production (R.H.S. of variable). If the PDA reads a terminal a on the PDS then erase it. If all the terminal erases by putting the production of variable them languages generated by grammar is not empty. If variable not converted into any production, means terminal string PDA will halts. In this case language generated by the grammar G is empty.

Q. 5. Attempt any two parts of the following : **$10 \times 2 = 20$**

(a) (i) Design a turing machine for the following language

$L = \{ww \mid w \in (a + b)^*\}$

Ans. A turing machine $(Q, \Sigma, \Gamma, \delta, q, b, F)$ for $L = \{ww / w \in (a + b)^*\}$ is given below.

Transition table

	a	b	Y
$\to q_0$	YRq_1	YRq_1	–
q_1	aRq_1	bRq_1	YLq_2
q_2	aLq_3	bLq_3	YRq_4
q_3	–	–	–
q_4	–	–	–

Q. 5. (a) (ii) Prove that if a language L and its complement both are recursively enumerable then L is recursive.

Ans. Let M_1 and M_2 be two TMS such that $L = T(M_1)$ and $\bar{L} = T(M_2)$. we construct new two type TM M that simulates M_1 on one tape and M_2 on the other.

If the input string W of M is in L, them M_1 accepts w and We declare that M accept w. If $w \in \bar{L}$, then M_2 accepts w and we declare that M halts without accepting. Thus in both cases, M eventually halts. By the construction of M it is clear that $T(M) = T(M_1) = L$. Hence L is recursive.

Q. 5. (b) Design a turning machine which computes function $f : N \to N$ defined as $f(n) = 2^n$.

Ans. As given $f(n) = 2^n$ where $n \in N$

$2^1 = 2 = 2 \times 1$

$2^2 = 4 = 2 \times 2$

$2^3 = 8 = 2 \times 4$

$2^4 = 16 = 2 \times 8$

$2^5 = 32 = 2 \times 16$

We can calculate $f(n)$ in the form of multiplication. We will denote every decimal number as a unary rotation.

For example $2 \times 2 = 4$ will be denoted as :

11 * 11 = 1111 (4)

–	–	B	1	1	*	1	1	=	B–	–

Final transition table is

	Q / Z	1	*	X	Y	=	B
→	q_0	q_1, x, R	$q_1, *, L$	–	–	–	–
	q_1	$q_1, 1, R$	$q_2, *, L$	–	–	–	–
	q_2	q_3, Y, R	–	–	–	$q_6, =, L$	–
	q_3	$q_3, 1, R$	–	–	–	$q_4, =, R$	–
	q_4	–	–	–	–	–	$q_5, 1, L$
	q_5	$q_5, 1, L$	–	–	q_2, Y, R	$q_5, =, L$	–
	q_6	–	$q_6, *, L$	$q_0, *, R$	$q_6, 1, L$	–	–
	q_7	–	–	$q_7, 1, L$	–	–	Accept

Q. 5. (c) Write short notes on the following :

(1) Universal turing machine

(2) Post correspondence problem.

Ans. Universal Turing Machine : A general purpose turing Machine is usually called universal turing machine which is powerful enough to simulate the behaviour of any computer, including any turing machine itself. More precisely universal turing machine can simulate the behaviour of an arbitrary turing machine over any Σ.

This turing's greatest discovery that there is a turing machine which is so sensitive in the respect that by properly adjusting the I/P representation, we can cause it to computer any turning computable function *i.e.*, any other function that can be computed by any other turing machine. The turing machine of the above capability is known as "UNIVERSAL TURING MACHINE".

5. (c). (2) The Post Correspondence Problem : The post correspondence problem (PCP) was first introduce by Emil Post in 1946. Later, the problem was found to have many application in the theory of formal languages. The problems and is stated as follows :

Consider the two list $x = (x_1 \ldots x_n)$, $y (y_1 \ldots y_n)$ of non empty strings over an alphabet, $\Sigma = \{0, 1\}$.

The PCP is to determine whether or not there exist $i_1, \ldots$, in where $1 \le ij \le n$, such that $x_{i_1} \ldots x_{i_m} = y_{i_1} \ldots y_{i_1}$

Example : Let $x = (b, ab^3, ba)$ and $y = (b^3, ba, a)$ have a solution ?

Ans. We have to determine whether or not there exist and the string formed by the sequence of corresponding substrings of y are identical. The require sequence is given by $i_1 = 2, i_2 = 1, i_3 = 1, i_4 = 3$ i.e. (2, 1, 1, 3), and $n = 4$. The

Corresponding strings are :

Thus the PCP has a solution.