Set – D

1) (i) 'c' (mark-1)

To Prove that the language is not regular

(ii) 'd' (mark-1)

$S \rightarrow BA \mid AB$

$A \rightarrow 0 \mid 1 \mid \dots 9$

$B \rightarrow A \mid B \mid \dots \mid z \mid a \mid b \mid \dots z.$

(iii) (mark-3)

Terminals

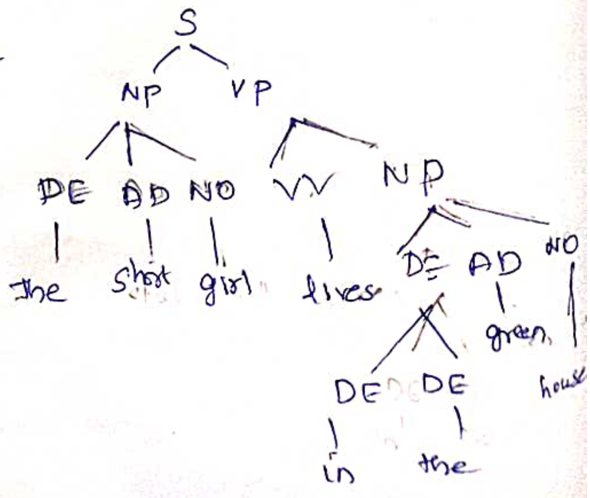
T = { the, a, in, with, red, short, tall, green, lives, swings, walks, girl, boy, game, dress, home }

Non terminals

V = { S, NP, VP, DE, AD, VV, NO }

(iv) Derive the ~~string~~ parse tree for the string. (mark 3)

" The short girl lives in the green house



S → NP VP

→ DE AD NO VP

→ The AD NO VP

→ The Short NO VP

→ The short girl VP

→ The short girl VV NP

→ The short girl lives NP

→ The short girl lives DE AD NO

→ The short girl lives DE DE AD NO

→ The short girl lives in DE AD NO

→ The short girl lives in the AD NO

→ The short girl lives in the green NO

→ The short girl lives in the green house.

→ The short girl lives in the green

(mark-5)

(v) The given grammer cannot generate the string "The tall boy hided in the house" either ambigously or unambigously because "hided" ~~in the string~~ terminal is not found in grammer.

(mark-12)

(vi) Given CFG

S → NP ~~V~~P

NP → AD NO | DE NO | DE AD NO |
NO

VP → V~~V~~ ~~N~~P

DE → DE DE | the | a | is | win.

AD → red | short | tall | green.

VV → lives | swings | walk .

NO → girl | boy | game | dress |

house .

CNF

A grammer is said to be in chomsky normal form if it has production of form

NT → NT . NT

NT → T .

simplification of CFG before converting to CNF .

1> the given grammar does not contain any ~~useful~~ useless symbol .

2> The given grammer does not contain 'ε' productions .

3> Elimination of unit productions.

Eleminating unit productions .

NP → NO we have .

S → NP VP

NP → AD NO | DE NO | DE AD NO

| girl | boy | game | dress |

house .

VP → VV NP

DE → DE DE | the | a | in | with

AD → red | short | tall | green.

VV → lives | swings | walks.

NO → girl | boy | game | dress | house.

Now, converting to CFG we have

S → NP VP

NP → AD NO

NP → DE NO

NP → DE NP [replacing AD NO → NP]

NP → girl | boy | game | dress | house.

VP → VP NP

DE → DE DE

DE → the | a | in | with.

AD → red | short | tall | green

VV → lives | swings | walks

NO → girl | boy | game | dress | house.

In the above grammar all the productions are of the form

NT → NT · NT

NT → T.

Thus converted to CNF.

2) (i) 'b' (Mark-1)

Even length palindromes

(ii) 'c' (Mark-1)

Elimination of generating symbols.

(iii) LMD of "while a<b do c=d" (4-marks)

x = "while a<b do c=d"

$Stmt \xrightarrow{lmd}$ while cond do stmt

$\xrightarrow{lmd}$ while id Relop id do stmt

$\rightarrow$ while a Relop id do stmt

$\rightarrow$ while a < id do stmt

$\rightarrow$ while a < b do stmt

$\rightarrow$ while a < b do id = Expr.

$\rightarrow$ while a < b do c = Expr

$\rightarrow$ while a < b do c = id

$\rightarrow$ while a < b do c = d.

(iv) RMD of "If a > b then c = d * e"
(4 marks)

$Stmt \xrightarrow{rmd}$ If cond then stmt

$\rightarrow$ If cond then id = Expr

$\rightarrow$ If cond then id = id op id

$\rightarrow$ If condn then id = id op e

$\rightarrow$ If cond then id = id * e

→ if cond then id = d * e

→ if cond then c = d * e

→ if id relop id then c = d * e

→ if id relop b then c = d * e

→ if id > b then c = d * e

→ if a > b then c = d * e.

Thus lmd is achieved.

(V) (Mark - 5)

PT the grammar is ambiguous grammer using string if a <= b then if c >= d then e = f else f = e

Defn

If a grammer is having 2 lmd or 2 rmd then that grammer is ambiguous grammer.

I lmd

Stmt $\xrightarrow{lmd}$ if condn then stmt else stmt.

→ if id relop id then stmt else stmt

→ if a relop id then stmt else stmt

→ if a <= id then stmt else stmt.

→ if a <= b then stmt else stmt.

→ if a <= b then if cond then stmt else stmt.

→ if a <= b then if id relop id then stmt else stmt.

→ if a <= b then if a relop id then stmt else stmt.

→ If a <= b then if c >= id then
stmt else stmt

→ if a <= b then if c >= d then stmt
else stmt

→ if a<=b then if c >= d then id = expr
else stmt

→ if a <= b then if c >= d then e = expr
else stmt

→ if a<=b then if c >= d then e = id
else stmt.

→ if a<=b then if c >= d then e = f
else stmt.

→ if a<=b then if c>=d then e = f
else id = expr

→ if a<=b then if c >= d then e = f
else f = expr

→ if a<=b then if c>=d then e=f
else f = id.

→ If a <= b then if c >= d then e=f
else f = e.

The string is derived using lmd → ①.

II - lmd

lmd.
stmt → if condn then stmt.

→ if id relop id then stmt

→ if a relop id then stmt

→ if a <= id then stmt

→ if a <= b then stmt.

→ If a<=b then if condn then stmt else stmt

→ if a<=b then if id relop id then stmt else stmt.

→ if a<=b then if c relop id then stmt else stmt.

→ if a<=b then if c >= id then stmt else stmt.

→ If a<=b then if c>=d then stmt else stmt.

→ If a<=b then if c>=d then id = Expr else stmt

→ if a<=b then if c>=d then e = expr else stmt.

→ If a<=b then if c>=d then e = id else stmt

→ If a<=b then if c>=d then e = f else stmt.

→ If a<=b then if c>=d then e = f else id = Expr

→ If a<=b then if c>=d then e=f else f = expr

→ If a<=b then if c>=d then e=f else f = id

→ If a<=b then if c>=d then e=f else f=e.

The same string is derived using II lmd. The above grammer is having 2 lmd for the given string so it is ambiguous grammer.

(vi) CFG to PDA. (7 marks)

~~Stmt → if condn~~

PDA P that accepts L(G) by empty stack is as follows.

1\ ~~Pro~~ $P = (\{q\}, T, V \cup T, \delta, q, S)$

where δ is defined as:-

1> For each variable A_i

$$\delta(q, \epsilon, A) = \{(q, \beta)\}$$

where $A \to \beta$ is a producti

2> For each nonterminal a

$$\delta(q, a, a) = \{(q, \epsilon)\}$$

Stmt → if condn then stmt |
if condn then stmt else stmt |
while condn do stmt |
id = Expr | id.

condn → id relop id

Expr → id op id | id

Relop → < | > | <= | >= | == | !=

op → + | - | * | / | %.

id → a | b | f | e | f.

PDA .

$\delta(q, \epsilon, stmt) = \{(q, \text{if condn then stmt}),$
$(q, \text{if condn then stmt else stmt}),$
$(q, \text{while condn do stmt}),$
$(q, id = Expr), (q, id)\}$

$\delta(q, \epsilon, condn) = \{(q, \text{id relop id})\}$

$\delta(q, \epsilon, Expr) = \{(q, \text{id relop id}), (q, id)\}$

$\delta(q, \epsilon, Relop) = ((q, <), (q, >), (q, >=)$
$(q, <=), (q, ==) (q, !=)\}$

$\delta(q, \epsilon, op) = \{(q, +), (q, -), (q, /), (q, \%)$
$(q, *)\}$

$\delta(q, id) = \{(q, a)\ (q, b)\ (q, c)\ (q, d)$
$(q, e)\ (q, f)\}$

$\delta(q, <, <) = (q, \epsilon)$
$\delta(q, >, >) = (q, \epsilon)$
$\delta(q, >=, >=) = (q, \epsilon)$
$\delta(q, <= <=) = (q, \epsilon)$
$\delta(q, ==, ==) = (q, \epsilon)$
$\delta(q, !=, !=) = (q, \epsilon)$
$\delta(q, +, +) = (q, \epsilon)$
$\delta(q, --) = (q, \epsilon)$

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

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

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

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

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

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

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

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

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

$\delta(q, \epsilon, \epsilon) = (q, \epsilon)$ // accepted empty state

(viii)

A string a accepted by a PDA if it has a path from initial to final state

Eg, ~~E,~~ a=b|c

stmt → id = Expr

⇒ a = Expr

→ a = id op id

→ a = b op id

→ a = b|id

→ a = b|c.

state	i/p	stack
q_0	a = b|c	Empty
q_0	a = b|c	stmt
q_0	a = b|c	id = Expr
q_0	a = b|c	a = Expr
q_0	= b|c	= Expr
q_0	b|c	Expr
q_0	b|c	id op id
q_0	b|c	b op id
q	|c	op id
q	|c	| id
q	|c	id
q	c	c
q	c	ε // accepted empty stack
q	ε	

The above string is accepted by PDA.

3) (i) No change (1 Mark)

(ii) Both ~~lang~~ languages are equal (1 mark)

(iii) Let x = MCQ

y = Fill up the blanks

z = Match the following

k = Descriptive.

Conc(i) student should choose.

$x^n y^{2n} z^m k^m$.

we can design a PDA by pushing all x and checking for every x there are $2y$'s and then push z to check its equality with k.

	Q	I/P	stack	Q x operations.
1)	q_0	x	Z_0	(q_0, xZ_0)
2)	q_0	x	x	(q_0, xx)
3)	q_0	y	x	(q_1, ϵ)
4)	q_1	y	x	(q_2, ϵ)
5)	q_2	y	x	(q_1, x)
6)	q_2	z	Z_0	(q_3, zZ_0)
7)	q_3	z	z	(q_3, zz)
8)	q_3	k	z	(q_4, ϵ)
9)	q_4	k	Z_0	(q_4, ϵ)
10)	q_4	ϵ	Z_0	(q_5, Z_0)

(iv)

Case (ii) Student should choose

$x^n y^{3n} z^p k^{2p}$.

we can design a PDA by pushing all x and checking for every x there are $3y$'s and the push all z and check for every z there are $2k$'s.

	Q	i/p	stack	Q x operations.
				(q_0, xZ_0)
1)	q_0	x	Z_0	(q_0, xx)
2)	q_0	x	x	$(q_1, \cancel{x})$
3)	q_0	y	x	(q_2, x)
4)	q_1	y	x	(q_3, ϵ)
			x	(q_1, x)
5)	q_2	y	x	(q_4, zZ_0)
6)	q_3,	y	Z_0	
		z		(q_4, zz)
7)	q_3		z	
		z		(q_5, ϵ)
8)	q_4	k	z	
				(q_6, ϵ)
9)	q_4		z	(q_5, z)
		k	z	(q_7, Z_0)
10)	q_5	k		
	q_6	ϵ	Z_0	
11)	q_6			accepting state,

(v) PDA

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

Q = set of states.

Σ = set of i/p symbols

q_0 - initial state.

δ : transistion function

Γ : stack symbols

z_0 - bottom of stack

F - final state.

$$P(\{q_0, q_1, q_2\}, \{x, y, z, k\}, q_0, \delta, \{\cancel{x, y, z,} k\}^{x, z\}}, z_0, \{q_2\})$$

↳ Formal defn of constructed PDA.

(vi) Empty stack PDA.

(i) iv → problem replace ~~line~~ transistions 10 with

$$\delta(q_4, \epsilon, z_0) = (q_5, \epsilon)$$

// (empty stack)

(ii) (v) → problem replace transistions 11 with

$$\delta(q_6, \epsilon, z_0) = (q_7, \epsilon)$$

// empty stack.