CT-1

CSPC41 - Compiler

1.) A Queue is a FIFO data structure which does not provide direct access to its data members.

→ Moreover, to parse through a queue, we would require another data structure to store the popped out {Key, value} pairs, thereby greatly increasing the accesstime and also the Space Complexity.

→ Thus, we CANNOT use a queue for symbol table.

2.) C code:-

printf (" The sum of %d numbers is %d", n, sum);

So, the tokens would be:-

1.) printf
2.) (
3.) " The sum of %d number is %d" → String literal
4.) ,
5.) n
6.) ,
7.) sum
8.))
9.) ;

→

5)

a) <u>Single line comment</u>:

$d_1 \rightarrow$ `//.*`

Example:- // This code prints sum.

b) <u>Multiline comment</u>:

→ Here we need to take care of line breaks too.

<u>Step 1</u>:- Consider the end-points `/* ..... */`

A ⇒ `/\*.*\*/`

Now, we need to understand the middle part.

<u>Step 2</u>:- Completing the internal part:-

B ⇒ `([^*]|[\r\n]|(\*+([^*/]|[\r\n])))*`

This part becomes regex for internal part., so, we call it d_2.

<u>Step 3</u>: Final Regex:-

d_3 ⇒ `/\*.d2.*\*+/`

c) <u>Combining both, we get</u>:-

→ $\boxed{d_4 \rightarrow d_3 \mid d_1}$ → <u>Ans</u>

4) Grammar :-

$S \rightarrow Tc \mid Sc \mid c$

$T \rightarrow Ta \mid Sa \mid d$

$C \rightarrow St \mid Td$

A- LEADING (A) = $\{a \mid A \Rightarrow \gamma a \delta\}$ → First terminal in str[ing]

TRAILING (A) = $\{a \mid A \Rightarrow \gamma a \delta\}$ → Last terminal in string

Terminals ⇒ $\{a, c, d, t\}$

Non-terminals = $\{S, T, C\}$

So, Leading (S) = $\{c, t, d, a\}$

Leading (T) = $\{a, d, t, c\}$

Leading (C) = $\{t, d, c, a\}$

Now, for trailing.

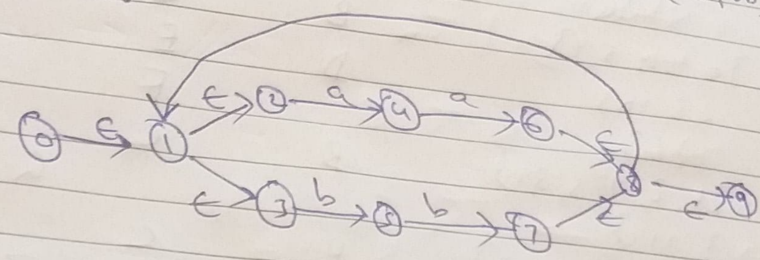
Trailing (S) = $\{c, t, d\}$

Trailing (T) = $\{a, d\}$

Trailing (C) = $\{t, d\}$

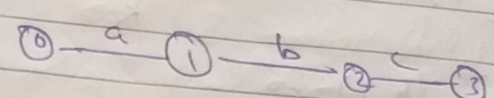
5) NFA for $(aa|bb)^* abc((ba|ca)(cc|bb)^*)^*$

$(aa|bb)^*$



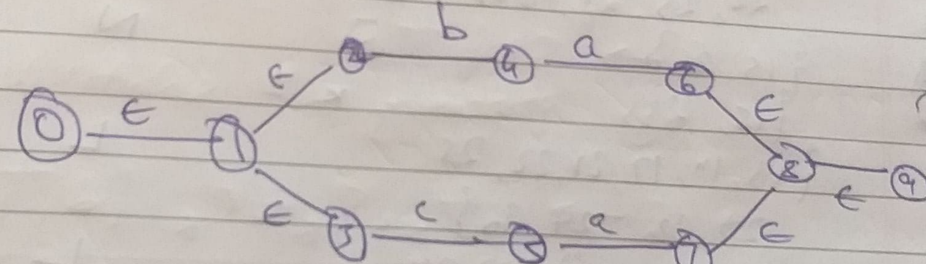
} N_1

abc



} N_2

$(ba|ca)$

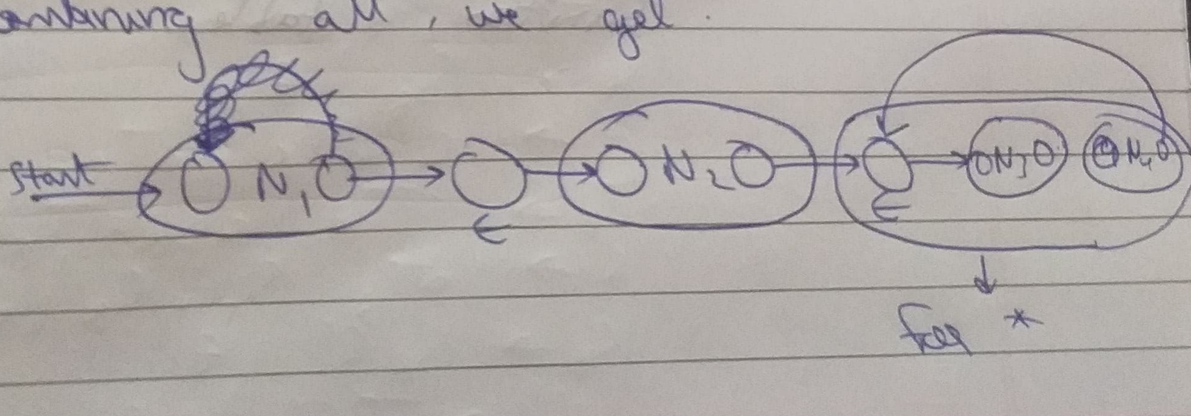


} N_3

$(cc|bb)^*$

Same as $(aa|bb)^*$ with $c \to a$, $b \to b$ } N_4

Combining all, we get.



Now, for conversion, we make state table.

	State	a	b	c
A	{0, 1, 2, 5, 8}	B	C	
B	{3, 10}	D	E	
C	{6}		F	
D	{1, 2, 4, 5, 8, 9}	B	C	G
E	{11}		G	
F	{1, 2, 5, 7, 8, 9}	B	H	I
G	{12, 13, 14, 17, 30}	J		
H	{15}	K		
I	{18}		L	M
J	{13, 14, 16, 17, 20, 21, 22, 25, 29, 30}		L	M
K	{13, 14, 17, 19, 20, 21, 22, 25, 29, 30}		N	
L	{15, 26}	J		O
M	{18, 23}	K	L	M
N	{13, 14, 17, 21, 22, 25, 27, 28, 29, 30}		L	M
O	{13, 14, 17, 21, 22, 24, 25, 28, 29, 30}			

The DFA can be constructed by simply following the above transition table.

Start
↓

6.) $R \to (R) \mid cT$

$T \to +R \mid *R \mid \epsilon$

	First	Follow
R	{ (, c }	{), $ }
T	{ +, *, ε }	{), $ }

Parsing Table

	(	)	c	+	*	$
R	$R \to (R)$		$R \to cT$			
T				$T \to +R$	$T \to *R$	$T \to \epsilon$

Let string :- ~~c*~~ c+ (c*c)

Start	Input	Action
$R	c+ (c*c) $	$R \to cT$
$Tc	c* (c*c) $	matching pop
$T	~~*~~+((c*c) $	$T \to \cancel{*}+R$
$R+	+ ((c*c) $	matching pop
$R	(c*c) $	$R \to CR$
$)R(	((c*c) $	matching pop

Similarly, we can match the internal c*c also.

Accepted

7) $S \to Tc \mid Sc \mid C \to LR$

$T \to Ta \mid Sa \mid d \to LR$

$C \to St \mid Td$

Ordering S, T, C

i) $S \to T_c S_R \mid C S_R$

$S_R \to c S_R \mid \epsilon$

ii) $T \to T_a \mid S_a \mid d$

Substituting $S \to T_c S_R \mid C S_R$

$T \to T_a \mid T_c S_R a \mid C S_R \mid d$

$T \to C S_R T_R \mid d T_R$

$T_R \to a T_R \mid c S_R a T_R \mid e$

iii) $C \to St \mid Td$

by processing & substituting,

$C \to d T_R c S_R t C_R \mid d T_R d C_R$

$\therefore$ $\boxed{\begin{array}{l} C' \to T_R C'' \mid t C_R \\ C'' \to c S_R t C_R \mid d C_R \end{array}}$

So, Final Productions:-

$S \to T_c S_R \mid C S_R$

$S_R \to S_R \mid \epsilon$

$T \to c S_R T_R \mid d T_R$

$T_R \to a T_R \mid c S_R a T_R \mid \epsilon$

$C \to d T_R c S_R t C_R \mid d T_R d C_R$

$C_R \to S_R C' \mid \epsilon$

$C' \to T_R C'' \mid t C_R$

$C'' \to c S_R t C_R \mid d C_R$