21/02/22

CSPC62 - CT1
Compiler

106119100
Rajneesh

Question (1)

(i) Tokens :
Token is a sequence of character that can be treated as a single logical entity.

Types of token : identifier, keywords, operators special symbol, constans.

~~word~~

(ii) Patterns :
A set of string in the input for which the same token is produced as o/p.
This set of strings is described by a rule called pattern associated with token

(iii) Lexeme :
A lexeme is a seq. of char. in the source program that is matched by the patten ~~of~~ for a token.

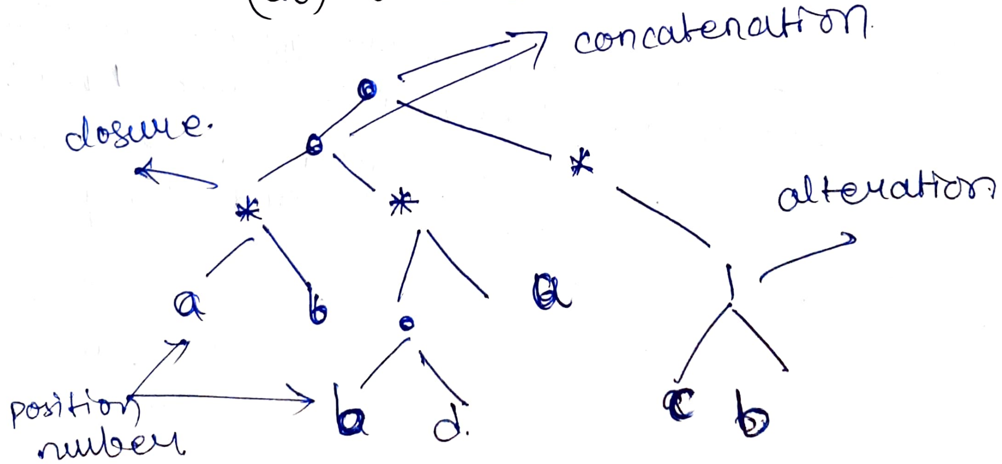
for example :

token	Lexeme	Pattern
ID	x y n1	letter followed by letters and digit
NUM	-123, 242	any numeric const
IF	if	if
LPAREN	(	(
LITERAL	"Hello"	any string of character (exect ") between " and "

Quest ②

regular expr.

$(ab)^* bda^* (c|b)^*$



DFA using the syntax tree.

Question 6

Given $S \rightarrow SbC \mid CeT \mid Se$ — LR

$T \rightarrow TaS \mid dC \mid a$ — LR.

$C \rightarrow St \mid Td$

Eleminating left reccursion ;

as, $C \rightarrow St \mid Td$. so.

Now, after removing left reccursion occur by S, T.

Introducing new nonterminal. and ordering S, T, C.

$S \rightarrow SbC \mid CeT \mid Se$

$S_R \rightarrow bC\,S_R \mid e\,S_R \mid \epsilon$

Now,

$T \rightarrow TaS \mid dC \mid a$

$T_R \rightarrow a\,S\,T_R \mid \epsilon$

Now, final grammer

$$\boxed{\begin{aligned} S &\rightarrow CeTS_R \\ T &\rightarrow dC\,T_R \mid a\,T_R \\ C &\rightarrow dC\,T_R\,d\,C_R \mid a\,T_R\,d\,C_R \\ S_R &\rightarrow bCS_R \mid eS_R \mid \epsilon \\ T_R &\rightarrow aST_R \mid \epsilon \end{aligned}}$$

Removing left factoring.

as, $S \rightarrow Sbc \mid CeT \mid Se$

so, $S \longrightarrow SS_R \mid CeT$

Given $T \longrightarrow TaS \mid dC \mid a$

$C \Longrightarrow St \mid Td$

and

$S_R \longrightarrow bc \mid e$.

Question (7):

LL pausing table for given gramerr,

First (S) $\Rightarrow$ { * , c , # }

First (A) $\Rightarrow$ { c , # }

First (B) $\Rightarrow$ { % ,) }

Follow (S) $\Rightarrow$ { \$, # , * , (}

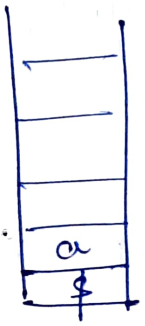
Follow(A) $\Rightarrow$ { # , * , (}

Follow (B) $\Rightarrow$ { \$, # , * (}

Parsing table:

	#	a	*	b	c	%	)	S
S	$S \to A\#a$		$S \to *bB$		$S \to A\#a$			
A	$A \to \#$				$A \to cAS$			
B						$B \to \%B$ #	$B \to)$	

Eg:



Stack.

string : a*b*b.

input = a*b*b$

Question ⑤ By the given grammer,

First (S) = {a} as $\underline{S \to aBcbD}$ Follow (S) = {b, a, h}

First (B) = {b, d} $\underline{B \to bBh \mid BD \mid d.}$ Follow (B) = {a, h, b}

First (C) = {a} Follow (C) = {b, a, h}

First (D) = {b, e, a} Follow (D) = {a, h, b}

First (E) = {a, e} Follow (E) = {a}

Question ⑧

	First	Follow
S	{ Elist], id }	{ $ }
E	{ (E) , Elist], id }	{ $, + }
L	{ Elist], id }	{ :, $, + }
Elist	{ Elist ,, id }	{ undefined }

Now LR table.

State	Action							Goto			
	:	=	+	(E)	Elist]	id	Elist[	S	E	L	Elist
0					s2	s3				1	
1	s4						r4				
2	r4		r4				r5				
3	r5		r5								
4		s5		s7	s2	s3					
5									6	8.	
6			s9				acc				
7			r2				r2				
8			r3				r3				
9											
10			s9 / r1				r1		10	8	

string str = ~~[illegible]~~
or
id + id * id.

Step	Stack	Input	Action
1	0	id + id *id $	s3
2	0 id 3	+ id * id $	r5
3	0 L	+ id * id $	1
4	0 L 1	+ id * id $	

Question 3

```
%{  # include <stidio.h>
    # include <string.h>
    int i;

%}

%%
[a-z  A-Z ]* {
  for (i=0 ; i<yyleng ; i++) {
  i    if ( yytex [i]='s' && yytex [i+1]='t'
                             && yytex[i+2]=r]
       return true.

} } [\t] *  return if (true)      yylex()
   .*  [ECHO]   print (yes) .     main()
                                  {    }
```