```
Algorithm for predictive parsing program -
input: A string wo and a parsing table M for Grammar G
output: if w is in L(GI), a left most derivation of w;
      otherwise an error Endication.
Method: Puitialty, the parser is fu a contiguration with.
 w$ fu the Euplit buffer and the start symbol 5 of G
 on top ob stack, above $.
         let 'a' be the first symbol of w;
          let X be the top stack symbol;
         while (x + 4)
          if (x = a) pop the stack and let a be next symbol obus,
        elseif (X 15 a terminal) error ();
      eself (MEX, a) is an error entry) error;
     elserf (M [x, a] = x -) Y1Y2 -- YK) 2
                output the production X -> YIY2 -- Yx;
                 pop the stack;
                  push YK, YK-1 - - YI on to the Stack, with
                                       You top;
    let x be the top stack symbol;
    Cost Motolin
Example- consider the bollowing grammar:
     E TEL
      E + TEIG
       T -> FT/
       TI - KETIE
        F > (E) | id
conspect predictive parsing table and also parse the string.
                                     id + id * id
```

5014, we have already constructed the predictive parsing table bor this grammar. Now we will parse inputsting id + id # id

	lid	1+	(	()	*	\$	OJ TI	cutbut:
XIII IOI	E-)TE	citions	E->TE		100,		erw rana	0
E	Jedonie	E-> +TE	0.81	El->C	-000C	Elac	Fall to	Method:
T	T-) FTI		T-)FT		4	4 6 63		9 400
T		T+E	o (= a = )		7-1-1	7176	te do d	OF NO
F	Fold	· )aa	F-)(E)	V + 2	1+ 31	had C		
\* 184m	FON A	Nonla .		Pre C	大 以	4 9d X	3.91	

Predictive parsing table

MATCHED		0 00	1
TIATCHED	STACK	IMPUT	ACTION
	E\$ Ophi	edfid*id\$	ETTE
nome	TE \$	1d + 1d * 1d\$	7 > FT
2 1	FTE \$	id + id * id\$	Faid
4 /	L'ates.	id + id * id\$	Match id
id W sy	TIELS NOTOUS	\$ 12 × 12 × 12 × 12 × 12 × 12 × 12 × 12	TAE
ld	E \$	+ 1d * 1d\$	E + TE
id to sopled	LTE'S	+ id * id\$	Match 't'
lid +	TEID	id # id \$	T> FT
id t	FTE \$	· ld # rd \$	Fald
	IN TES	· 12 * 12 \$	Match 1'd
181 +	TIEID	. * rd\$	1 -> * ET
id + id	*FT ES	* id\$	match' *
ld + ld		12 \$	F → i'd
d'd +1'd *	FT ES	ed \$	Model 1'd
rd + rd +k	id TES		7126
id + 1d * 1d	TES	\$	
	= = =	\$ 1	E-) C
id + id * id		\$ 511 (;	Accept
id + id * id	parnd od n bn	· P · · · · · · · · · · · · · · · · · ·	ricepe
		THE DOOR SYN	DILLON House

- 7 Predictive parsers can be constructed for a class of Grammans, carled LL(1). The birst "L" stands bor scanning the Input brom left to right. The second "L" for producing a left most derivation and "1" bor using one input symbol of rookahead at each step to make parsing action decisions.
- -> No lest recursive / Non Determinetic or antiguous granuar can be LL(1).
- + For every LL(1) grammar, each parsing table entry uniquely identifies a production.

NOTE- if en the question et is indicated that you have to construct Predictive parsing table them before constructing predictive parsing table, eliminale ambiguity, lettremeni and non determination is En the question you have to check. bor LL(1) grammar then no need to eliminate ambiguly, left recursion or Non determination grammar.

example- consider the bollowing grammar SA L'Etss/a s - es E

check that the grahemar is LLII) or not.

check the	see The			-101-1	0'. 02 FC	LLOW(S)=(e,\$)
301h	S - LE	tss   a	FIRS	T(S) = h T(S) = h		10m(3)=[e,\$]
	sl 7 es			T(E) = {		10W(E) = f.t3
1.	$E \rightarrow b$	, 1	b	e	4 11/11	multiple detired
3	salietss t	Sta				entriel, so
SI				shes)	SHE	not LLU)
						1 (0.2

- (1) s asbs wown left to right "The second "L" for bro bsas in the destruction and "1" box using one input of
- S- aABb (2) A NO left recursive I Novi Determination or a A - CIE B - dIE
- S-) aB/E B  $\rightarrow$  b C [  $\in$  4 below 26 to included by 1) 2 d  $\leftarrow$  3 CACS EN work state greened avidables danger sources
- bredictive parally table; eliminals ambiguity, (4) is so AB of AB A -) ale planuard of been on right remaining that most
- S-> A (5) A + Bb/Cd B-) aB | E C+ CCIE
- (6) S-) aAa| E A-) abs E

identifies a production.

ork that the granular is LLCD or het Awerws: (1) not LL(1)

- 1401-16(2) 14L(1) 01= (6)10917 3 20 6 12
- (# (= (3) (3) LL(1) (d) = (3) TESTS
  - (4) [(1)]
  - (5) LL(1)
- (6) not LL(1)