	C	hirag Dod	ria	
	0	12C 17	7	
	2	system Prog Exerciment N	ranning & Compiler construction	Lal
	-		ertation of Operator Precedence Pare	
	3	reory:-		
		<i>d</i>		
_	0	recator Prec	idence Pauser is a bottom up parse	er
_			yout an operator percedance gran	
_			ucidince paising technique can be	
	11		eator grammars	
-	00	water gram	nan is said to be context free Gramm	ar
	CG	FG) if that s	atisfies two conditions:	
			t contain any rull production	
	11		s should not appear together	
_			dence pauser ruly on the following	0
	th	un ruceder	ce relations:	χ
	1	90.1	with the same took of a contract to the district	
		Relation	Description	1
	٠.	IA < B	· A give puccedince to B	
		A>B	A takes puccedence over 8	
		A = B	A&B have equal pricedince	

Steps for clisigning Operator Precedence Parser

- Design precedence relation table or precedence relation
matrix for the given operator grammar

- Write precedence relation algorithm

- Give some examples

Example Design operator precedence passer for E-> E+E E × E id Step 1:- Design operator precedence relation talm id + × id - > > > + < > < > > × × ×							
Step 1:- Design operator precedence relation talk id + x \$ id - > > > > id - > > > > x < > > > > \$ x < > > > > > > \$ x < > > > > > > \$ x < > > > > > > > > \$ x < > > > > > > > > > > \$ x < > > > > > > > > > > > > > x < x & x & x & x & x & x & x & x & x &	Example						
id + x \$ id - > > > + < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X < > > X > > X < X > > > X > > X							
id - > > > > > > > > > > > > > > > > > >	Step 1:-	Dusign	орин	ato4	pucc	duce s	ulation table
id - > > > > > > > > > > > > > > > > > >			ico	+	X	\$	
# < > > > > > > > > > > > > > > > > > >		id	-	>	>	>	
Step 2:- Algorithm Repeat fourex K If Input is completely scarned & stack contain only start variable then accept & break Else Let 'a' be the topmost stack variable & b be the nort terminal If A < B or A = B Then Reduce		+	. <		, . < .	>	850.21 3 -
Step 2:- Algorithm Repeat forwer If Input is completely scarred & stack contain only start variable then occupt & break Else Let 'a' be the topmost stack variable & b be the input terminal If A < B or A = B Then Reduce		X	< ,			>	
Step 2:- Algorithm Repeat forever If Input is completely scanned & stack contain only start variable then accept & break Else Let 'a' be the topmost stack variable & b be the input terminal If A < B or A = B Then Reduce			<	<	. <	>	- This is no
Else Let 'a' be the topmost stack variable & b be the input terminal If A < B or A = B Then Reduce	ac			h 1254		- 1,0016 5,98 (531) 55 () (2	
Else Jet 'a' be the topmost stack variable & b be the "nout terminal If A < B or A = B Yhin shift Else if; A > B Then Reduce	Input is	comple	tely &	carre	l & s	tack co	ntain only
Let 'a' be the topmost stack variable & b be the input terminal If A < B or A = B Yhir shift Else if: A > B Then Reduce			,			restr.	.3.
is put terminal If A < B or A = B Yhin shift Else if: A > B Then Recluce		be the	torno	st sto	rck va	rialde	& b be the
A < B or A = B Then Reduce				는 무리		4	5.
Else if: A > B Then Recluce				, v	Ý	<u>.</u>	`
Then Recluce	This shift	tores y	32000	0 0	500		
Then Reduce					1 1.13	dis me	a di sa
					1 - 3	<u> </u>	20 . •
	The Redu	CC CEC D	5-12	पर तर्दर			

Step 3:-	Example			
Stack	Input	А	В	Action
\$	id + id * id \$	\$	id	shift id
t id	+id *id\$	id	+	Reduce € -rid
E	+ id * id \$	\$	+	shift +
SE+	id * id\$	+	id	shift id
e+id	* id \$	id	*	Reduce € → id
E+E	* id\$	\$	*	Reduce € → € + €
ϵ	* id\$	\$	*	shift *
: E *	id \$	*	id	shift id
SE * id	\$	id	\$	Reduce E→id
E * E	\$	\$	\$	Reduce € → € * €
ϵ	\$	\$	\$	Accept