Page No.
Class
Roll No.
Date: / /202

Practical No. 05

Title :- Write a C program to implement operator precedence parsing.

Theory :- A grammer that is generated to define the mathematical operator is called operator grammer with some restriction on grammer ; An operator precedence grammer is a context free grammer that has the property that no production has either an empty right hand side or two adjacent non-terminal in its right hand side.

Operator precedence grammer is kind of shift reduce parsing method It is applied to a small class of operator grammers.

1. A grammers is said to be operator precedence grammar if it has two properties.
2. NO R.H.S. of any production has a ϵ.
3. No two non-terminals are adjacent.
4. Operator precedence can only estabilished between the terminal of the grammer. It ignores the non-terminals

There are the three operator precedence relations :-

- $a > b$ means that terminal "a" has the higher precedence than terminal "b".
- $a < b$ means that terminal "a" has the lower precedence than terminal "b".
- $a = b$ means that the terminal "a" and "b" both have same precedence

Page No.
Class
Roll No.
Date:

Precedence table

	+	*	(	)	id	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	x
)	>	>	x	>	x	>
id	>	>	x	>	x	>
$	<	<	<	x	<	x

Parsing Action :-

- Both end of the given input string add the $ symbol.
- Now scan the input string from left right until the > is encountered.
- Scan toward left over all the equal precedence until the first left most < is encountered
- Everything between leftmost < and right most > is handle
- $ on $ means parsing is successful

Ex:- Grammer

E → E + T / T

T → T * F / F

F → id

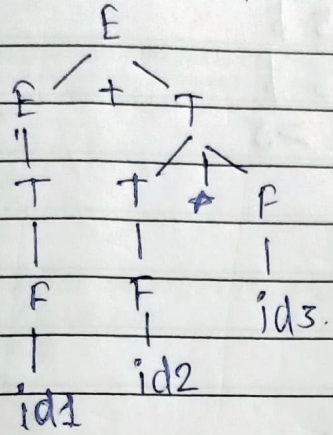
Given string

I = id + id * id.

Page No.
Class
Roll No.
Date: / /202

~~practic~~

Let us consider a parse tree for it as a follows



On the basis of the above tree we can design

	E	T	F	id	+	*	$
E	X	X	X	X	=	X	>
T	X	X	X	X	>	=	>
F	X	X	X	X	>	>	>
id	X	X	X	X	>	>	>
+	X	=	<	<	X	X	X
*	X	X	=	<	X	X	X
$	<	<	<	<	X	X	X

Now let us process the string with the help of the above precedence table

$ <id 1> + id2 * id3 $

$ <F> + id2 * id3 $

$ <T> + id2 * id3 $
$ <E = + <id2> * id3 $
$ <E = + <F> * id3 $
$ <E = + <T = * = f >S
$ <E = T = T >S
$ <E = + = T >S
$ <E >S
Accept.