Chirag Dodeja
D12C 17
System Programming & Compiler construction Lab
Experiment No

Aim :- Implementation of Operator Precedence Parser

Theory :-

- Operator Precedence Parser is a bottom up parser
- It is used to interpret an operator precedance grammar
- The operator precedence parsing technique can be applied to operator grammars
- Operator grammar is said to be context free Grammar (GFG) if that satisfies two conditions:
  a. It should not contain any null production
  b. Two variables should not appear together
- Operator Precedence parser rely on the following three precedence relations:

Relation	Description
A < B	A give precedence to B
A > B	A takes precedence over B
A = B	A & B have equal precedence

Steps for designing 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 parser for
$E \rightarrow E + E \mid E \times E \mid id$

Step 1 :- Design operator precedence relation table

	id	+	x	$
id	-	>	>	>
+	<	>	<	>
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 shift
Else if
        A > B
Then Reduce
}
```

Step 3 :- Example

Stack	Input	A	B	Action
\$	id + id * id\$	\$	id	shift id
\$ id	+ id * id \$	id	+	Reduce E → id
\$ E	+ id * id \$	\$	+	shift +
\$ E +	id * id \$	+	id	shift id
\$ E + id	* id \$	id	*	Reduce E → id
\$ E + E	* id\$	\$	*	Reduce E → E + E
\$ E	* id \$	\$	*	shift *
\$ E *	id \$	*	id	shift id
\$ E * id	\$	id	\$	Reduce E → id
\$ E * E	\$	\$	\$	Reduce E → E * E
\$ E	\$	\$	\$	Accept