Date: 25/09/2025

Experiment 3.3

AIM

To develop operator precedence parser for a given grammar

ALGORITHM

1. Start
2. Initialize input and output stack which return '$' as default value if stack is empty.
3. Read Grammar.
4. Read precedence table.
5. Read Input.
6. Traverse input string in reverser and add to stack.
7. While input isn't marked as valid or invalid, do the following:
    1. Let l and r be the top non terminals on output and input stacks respectively. Find action a to take based on precedence table for l and r.
    2. If a='>' or a='=', Try to reduce by doing the following:
        1. For each production of the form $X \rightarrow Y_1Y_2..Y_k$:
            1. Check if top k elements of the stack are $Y_k, ,Y_{k-1},\ldots,Y_1$.
            2. If the stack matches the expression, pop k elements on the stack.
            3. Add X to the stack.
            4. Add the production to the Right Most Derivation (in reverse).
            5. Reduction is successful. Therefore, exit for loop.
        2. If reduction is unsuccessful, mark string as rejected.
    3. Otherwise, if a='<':
        1. If input stack is not empty, Shift by popping and element from input stack and pushing it into the output stack.
        2. If shifting fails, mark string as rejected.
    4. Otherwise, if a='A':
        1. Check if input stack is empty and output stack has only a single symbol corresponding to the start symbol. If yes, string is marked as accepted.

2. Otherwise, string is marked as rejected.

8. Display sequence of steps taken and the RMD generated.
9. Stop

RESULT

Successfully implemented operator precedence parser for the given grammar.