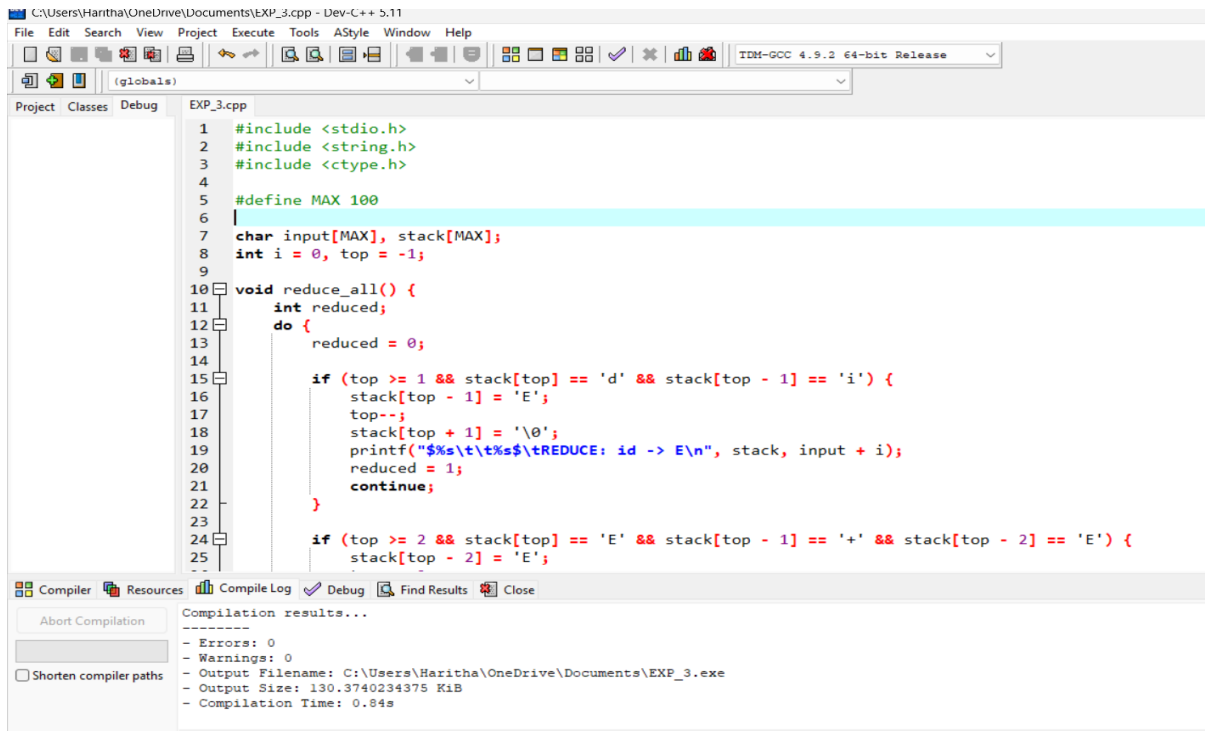


# EXPERIMENT-14

## AIM:

Implement the concept of Shift reduce parsing in C Programming.

## PROGRAM:

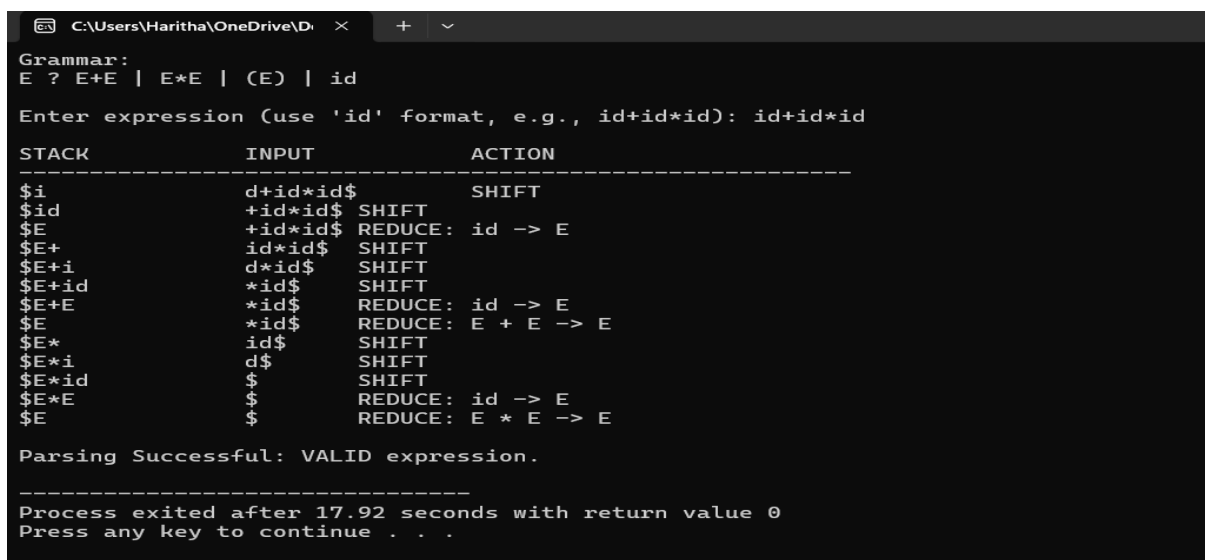


```
1 #include <stdio.h>
2 #include <string.h>
3 #include <ctype.h>
4
5 #define MAX 100
6
7 char input[MAX], stack[MAX];
8 int i = 0, top = -1;
9
10 void reduce_all() {
11     int reduced;
12     do {
13         reduced = 0;
14
15         if (top >= 1 && stack[top] == 'd' && stack[top - 1] == 'i') {
16             stack[top - 1] = 'E';
17             top--;
18             stack[top + 1] = '\0';
19             printf("$s\t\t%s$\tREDUCE: id -> E\n", stack, input + i);
20             reduced = 1;
21             continue;
22         }
23
24         if (top >= 2 && stack[top] == 'E' && stack[top - 1] == '+' && stack[top - 2] == 'E') {
25             stack[top - 2] = 'E';
26         }
27     } while (reduced == 0);
28 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Haritha\OneDrive\Documents\EXP\_3.exe
- Output Size: 130.3740234375 KiB
- Compilation Time: 0.84s

## OUTPUT:



```
Grammar:
E ? E+E | E*E | (E) | id

Enter expression (use 'id' format, e.g., id+id*id): id+id*id

STACK      INPUT      ACTION
-----
$i          d+id*id$    SHIFT
$id         +id*id$    SHIFT
$E          +id*id$    REDUCE: id -> E
$E+         id*id$    SHIFT
$E+i        d*id$      SHIFT
$E+id       *id$      SHIFT
$E+E        *id$      REDUCE: id -> E
$E          *id$      REDUCE: E + E -> E
$E*         id$      SHIFT
$E*i        d$        SHIFT
$E*id       $         SHIFT
$E*E        $         REDUCE: id -> E
$E          $         REDUCE: E * E -> E

Parsing Successful: VALID expression.

-----
Process exited after 17.92 seconds with return value 0
Press any key to continue . . .
```