CODE:

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

enum token {NUMBER, PLUS, MINUS, MULT, DIV, END, INVALID};

enum token current\_token;

int current\_value;

int error\_flag = 0; // Global flag to track errors

void print\_token(enum token t) {

switch (t) {

case NUMBER: printf("\nPARSER\_INTEGER---------(%d) ", current\_value); break;

case PLUS: printf("\nPARSER\_SYMBOL\_PLUS-----(+) "); break;

case MINUS: printf("\nPARSER\_SYMBOL\_MINUS-----(-) "); break;

case MULT: printf("\nPARSER\_SYMBOL\_MULT-----(\*) "); break;

case DIV: printf("\nPARSER\_SYMBOL\_DIV------(/) "); break;

case END: printf("\nEND\n"); break;

case INVALID: printf("\nINVALID\_TOKEN "); break;

}

}

void get\_token(char \*expr, int \*index) {

while (isspace(expr[\*index])) (\*index)++; // Skip whitespace

if (isdigit(expr[\*index])) {

current\_token = NUMBER;

current\_value = 0;

while (isdigit(expr[\*index])) {

current\_value = current\_value \* 10 + (expr[\*index] - '0');

(\*index)++;

}

} else {

switch (expr[\*index]) {

case '+': current\_token = PLUS; (\*index)++; break;

case '-': current\_token = MINUS; (\*index)++; break;

case '\*': current\_token = MULT; (\*index)++; break;

case '/': current\_token = DIV; (\*index)++; break;

case '\0': current\_token = END; break;

default: current\_token = INVALID; (\*index)++; error\_flag = 1; break;

}

}

if (current\_token != INVALID) {

print\_token(current\_token); // Print the token only if it's valid

}

}

void error\_recovery(char \*expr, int \*index) {

printf("\nError detected, skipping to next valid token...\n");

while (current\_token == INVALID) {

get\_token(expr, index); // Skip erroneous tokens

}

}

int term(char \*expr, int \*index);

int factor(char \*expr, int \*index);

int term(char \*expr, int \*index) {

int result = 0;

if (current\_token == NUMBER) {

result = current\_value;

get\_token(expr, index);

} else {

printf("\nSyntax error: Expected number, found ");

print\_token(current\_token); // Print invalid token

error\_flag = 1;

error\_recovery(expr, index);

result = 0; // Assign default value so calculation continues

}

return result;

}

int factor(char \*expr, int \*index) {

int result = term(expr, index);

while (current\_token == MULT || current\_token == DIV) {

if (current\_token == MULT) {

get\_token(expr, index);

result \*= term(expr, index);

} else if (current\_token == DIV) {

get\_token(expr, index);

int divisor = term(expr, index);

if (divisor == 0) {

printf("\nError: Division by zero! Assuming result as 1.\n");

error\_flag = 1;

result = 1; // Default value to prevent crashing

} else {

result /= divisor;

}

}

}

return result;

}

int expr(char \*expr, int \*index) {

int result = factor(expr, index);

while (current\_token == PLUS || current\_token == MINUS) {

if (current\_token == PLUS) {

get\_token(expr, index);

result += factor(expr, index);

} else if (current\_token == MINUS) {

get\_token(expr, index);

result -= factor(expr, index);

}

}

return result;

}

int main() {

char expr\_input[100];

printf("Enter an arithmetic expression using(\"+-\*/\"): ");

fgets(expr\_input, sizeof(expr\_input), stdin);

int index = 0;

get\_token(expr\_input, &index);

int result = expr(expr\_input, &index);

if (current\_token == END) {

if (error\_flag) {

printf("\nFinal\_result of the operation (with errors corrected): %d\n", result);

} else {

printf("\nFinal\_result of the operation: %d\n", result);

}

} else {

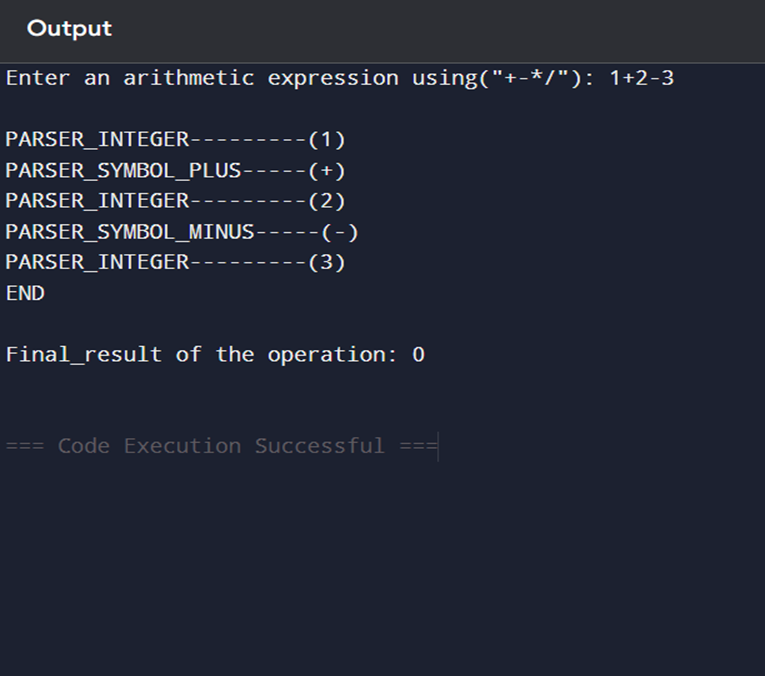
printf("\nError: Unexpected token at end of expression. Partial result: %d\n", result);

}

return 0;

}

**Output:**



Output 2:

