 Implement the concept of Shift reduce parsing in C Programming.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100

char stack[MAX];

int top = -1;

char input[MAX];

int position = 0;

// Push onto the stack

void push(char symbol) {

stack[++top] = symbol;

}

// Pop from the stack

void pop() {

top--;

}

// Display stack and input for debugging

void display() {

printf("Stack: ");

for (int i = 0; i <= top; i++)

printf("%c", stack[i]);

printf("\t Input: %s\n", &input[position]);

}

// Try to reduce the stack contents based on grammar rules

void reduce() {

while (1) {

// Reduction: id → E

if (top >= 1 && stack[top] == 'd' && stack[top - 1] == 'i') {

pop();

stack[top] = 'E'; // Replace 'i' with 'E'

continue;

}

// Reduction: E + E → E

if (top >= 2 && stack[top] == 'E' && stack[top - 1] == '+' && stack[top - 2] == 'E') {

pop();

pop();

continue;

}

// Reduction: E \* E → E

if (top >= 2 && stack[top] == 'E' && stack[top - 1] == '\*' && stack[top - 2] == 'E') {

pop();

pop();

continue;

}

// Reduction: (E) → E

if (top >= 2 && stack[top] == ')' && stack[top - 1] == 'E' && stack[top - 2] == '(') {

pop();

pop();

stack[top] = 'E'; // Replace '(' with 'E'

continue;

}

break; // No further reduction possible

}

}

// Shift-Reduce Parsing

void shift\_reduce\_parsing() {

printf("\nShift-Reduce Parsing Steps:\n");

while (position < strlen(input)) {

push(input[position++]); // Shift

display();

reduce(); // Try to reduce

display();

}

// Final check: The stack should contain only 'E'

if (top == 0 && stack[top] == 'E')

printf("\nParsing successful!\n");

else

printf("\nParsing failed!\n");

}

int main() {

printf("Enter an expression: ");

scanf("%s", input);

shift\_reduce\_parsing();

return 0;

}

**OUTPUT:**

