14)Implement the concept of Shift reduce parsing in C Programming.

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100

char stack[MAX][MAX];

int top = -1;

void push(char \*item) {

if (top < MAX - 1) {

strcpy(stack[++top], item);

} else {

printf("Stack overflow\n");

exit(1);

}

}

char \*pop() {

if (top >= 0) {

return stack[top--];

} else {

printf("Stack underflow\n");

exit(1);

}

}

void display\_stack() {

printf("Stack: ");

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

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

}

printf("\n");

}

int is\_terminal(char \*symbol) {

return strcmp(symbol, "id") == 0 || strcmp(symbol, "+") == 0 || strcmp(symbol, "\*") == 0 || strcmp(symbol, "(") == 0 || strcmp(symbol, ")") == 0;

}

void shift(char \*symbol) {

printf("Shift: %s\n", symbol);

push(symbol);

display\_stack();

}

void reduce() {

if (top >= 2 && strcmp(stack[top - 1], "+") == 0 && is\_terminal(stack[top - 2]) && is\_terminal(stack[top])) {

printf("Reduce: E -> E + E\n");

pop(); // id

pop(); // +

pop(); // id

push("E");

} else if (top >= 2 && strcmp(stack[top - 1], "\*") == 0 && is\_terminal(stack[top - 2]) && is\_terminal(stack[top])) {

printf("Reduce: E -> E \* E\n");

pop(); // id

pop(); // \*

pop(); // id

push("E");

} else if (top >= 2 && strcmp(stack[top - 2], "(") == 0 && is\_terminal(stack[top - 1]) && strcmp(stack[top], ")") == 0) {

printf("Reduce: E -> ( E )\n");

pop(); // )

pop(); // id

pop(); // (

push("E");

} else if (top >= 0 && is\_terminal(stack[top])) {

printf("Reduce: E -> id\n");

pop(); // id

push("E");

} else {

printf("Reduction not possible\n");

}

display\_stack();

}

void parse(char \*input) {

char \*token = strtok(input, " ");

while (token) {

shift(token);

reduce();

token = strtok(NULL, " ");

}

}

int main() {

char input[] = "id + id \* id";

printf("Input: %s\n", input);

parse(input);

return 0;

}

**Output:**

