```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#ifdef _WIN32
    #include <Windows.h>
#endif

char *
strsep(stringp, delim)
    register char **stringp;
    register const char *delim;
{
    register char *s;
    register const char *spanp;
    register int c, sc;
    char *tok;

    if ((s = *stringp) == NULL)
        return (NULL);
    for (tok = s;;) {
        c = *s++;
        spanp = delim;
        do {
            if ((sc = *spanp++) == c) {
                if (c == 0)
                    s = NULL;
                else
                    s[-1] = 0;
                *stringp = s;
                return (tok);
            }
        } while (sc != 0);
    }
}

int prefix(const char *pre, const char *str)
{
    return strncmp(pre, str, strlen(pre)) == 0;
}


typedef struct Rule {
    char* lhs;
    char* rhs;
} rule_t;



void parse_grammar(char* grammar_str, rule_t* rules, int* n_rules) {
    // printf("%s\n", grammar_str);
    char* ptr;
    char* temp = malloc(strlen(grammar_str) + 1);
    strcpy(temp, grammar_str);
    char * temp_lhs = malloc(strlen(grammar_str) + 1);
    char * temp_rhs = malloc(strlen(grammar_str) + 1);
    while ((ptr = strsep(&temp, "\n")) != NULL) {
        char* ptr2 = malloc(strlen(ptr) + 1);
        strcpy(ptr2, ptr);
        while((ptr = strsep(&ptr2, ":")) != NULL) {
            char* ptr3 = malloc(strlen(ptr) + 1);
            strcpy(ptr3, ptr);
            if(strstr(ptr3, "|") != NULL) {
```

```
                char* ptr4 = malloc(strlen(ptr3) + 1);
                strcpy(ptr4, ptr3);
                while((ptr3 = strsep(&ptr4, "|")) != NULL) {
                    // printf("RHS:%s\n", ptr3);
                    rules[*n_rules].lhs = strdup(temp_lhs);
                    rules[*n_rules].rhs = strdup(ptr3);
                    (*n_rules)++;
                }
            }
            else {
                // printf("LHS:%s\n", ptr3);
                strcpy(temp_lhs, ptr3);
            }
        }
    }
}

void print_grammar(rule_t* rules, int n_rules, int* rules_status) {
    for(int i = 0; i < n_rules; i++) {
        if (rules_status[i] == -1) continue;
        // printf("%s -> %s || %d\n", rules[i].lhs, rules[i].rhs,
rules_status[i]);
        printf("%s -> %s\n", rules[i].lhs, rules[i].rhs);
    }
}

unsigned short int check_left_recursion(rule_t rule) {
    if(prefix(rule.lhs, rule.rhs)) {
        return 1;
    }
    return 0;
}

int main(int argc, char *argv[]) {
    #ifdef _WIN32
        SetConsoleOutputCP(CP_UTF8);
    #endif

    char* grammar_str;

    if (!(argc < 2) && strcmp(argv[1], "-a") == 0) {
        grammar_str = "S:(L)|a\nL:L,S|S";
    } else {
        grammar_str = "E:E+E|E*E|id";
    }
    rule_t* rules = malloc(100 * sizeof(rule_t));
    int n_rules = 0;
    int rules_status[100] = {0};
    char * symbols_epsilons[100];
    int n_se = 0;
    parse_grammar(grammar_str, rules, &n_rules);

    printf("Original Grammar:\n");
    print_grammar(rules, n_rules, rules_status);

    for(int i = 0; i < n_rules; i++) {
        // printf("%d\n", check_left_recursion(rules[i]));
        if (check_left_recursion(rules[i])) {
            rules[n_rules].lhs = strdup(rules[i].lhs);
            strcat(rules[n_rules].lhs, "'");
            // while()
            // str_replace(rules[i].rhs, rules[i].lhs, "")
            rules[n_rules].rhs = strcat(rules[i].rhs, rules[n_rules].lhs);
```

```
            rules[n_rules].rhs++;
            rules_status[i] = -1;
            rules_status[n_rules] = 1;

            if (n_se == 0) {
                symbols_epsilons[n_se] = strdup(rules[n_rules].lhs);
                n_se++;
            }
            for (int j = 0; j <= n_se; j++) {
                // check if symbol is already present in symbols_epsilons
                char * temp = strcat(rules[i].lhs, "'");
                if (strcmp(symbols_epsilons[j], temp) == 0) {
                    break;
                }
                symbols_epsilons[n_se] = strdup(rules[n_rules].lhs);
                n_se++;
            }



            n_rules++;
        }
        else {
            if (rules_status[i] == -1) continue;
            if (rules_status[i] == 1) continue;

            for(int j = 0; j <= n_se; j++) {
                char * temp = strcat(strdup(rules[i].lhs), "'");
                if (strcmp(symbols_epsilons[j], temp) == 0) {
                    char * temp2 = strdup(rules[i].lhs);
                    rules[i].rhs = strcat(rules[i].rhs, strcat(temp2, "'"));
                    break;
                }
            }
        }
    }

    for(int i = 0; i < n_se; i++) {
        rules[n_rules].lhs = strdup(symbols_epsilons[i]);
        rules[n_rules].rhs = strdup("ϵ");
        rules_status[n_rules] = 3;
        n_rules++;
    }

    printf("Grammar after removing left recursion:\n");
    print_grammar(rules, n_rules, rules_status);

    return 0;
}
```