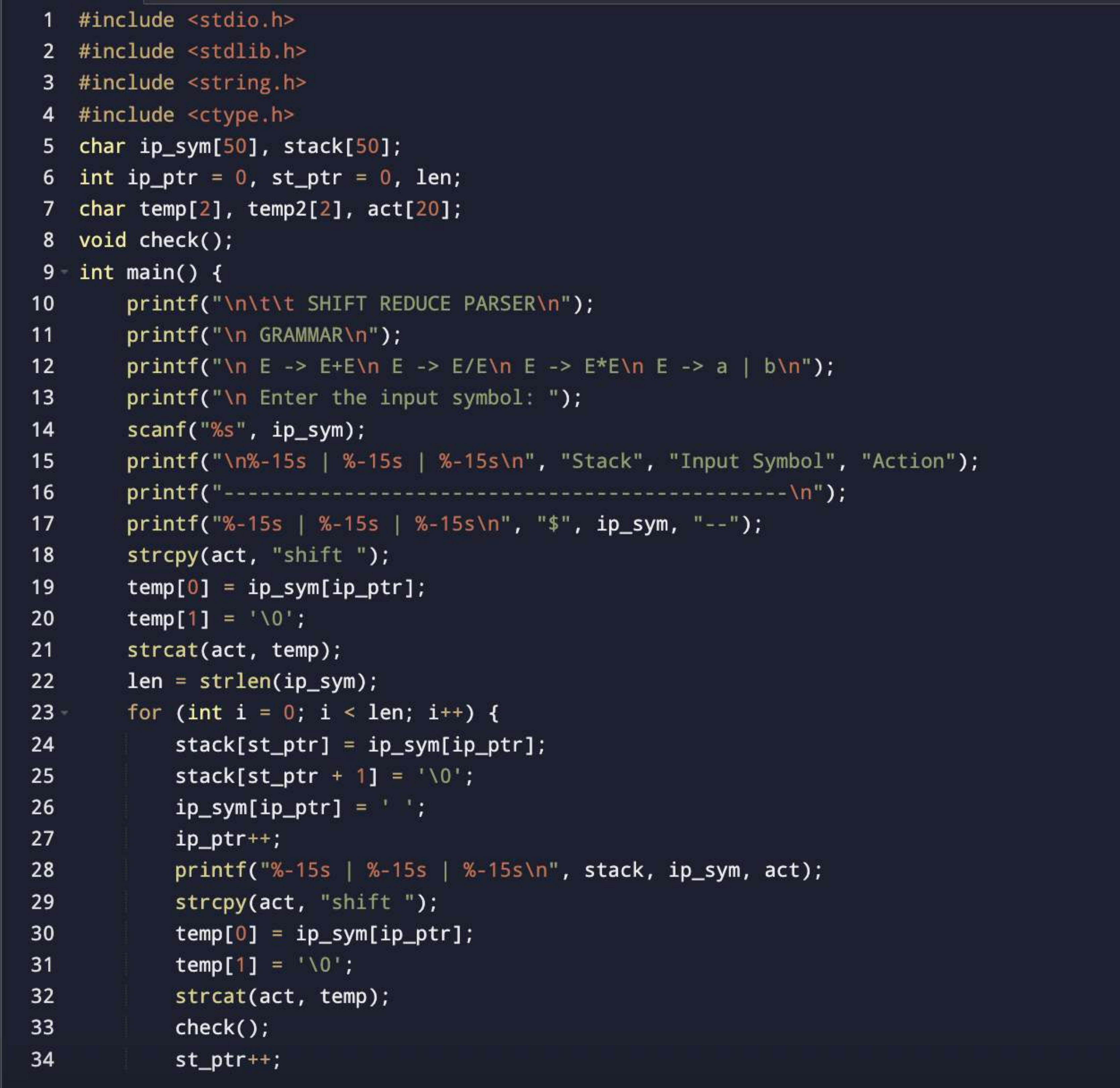
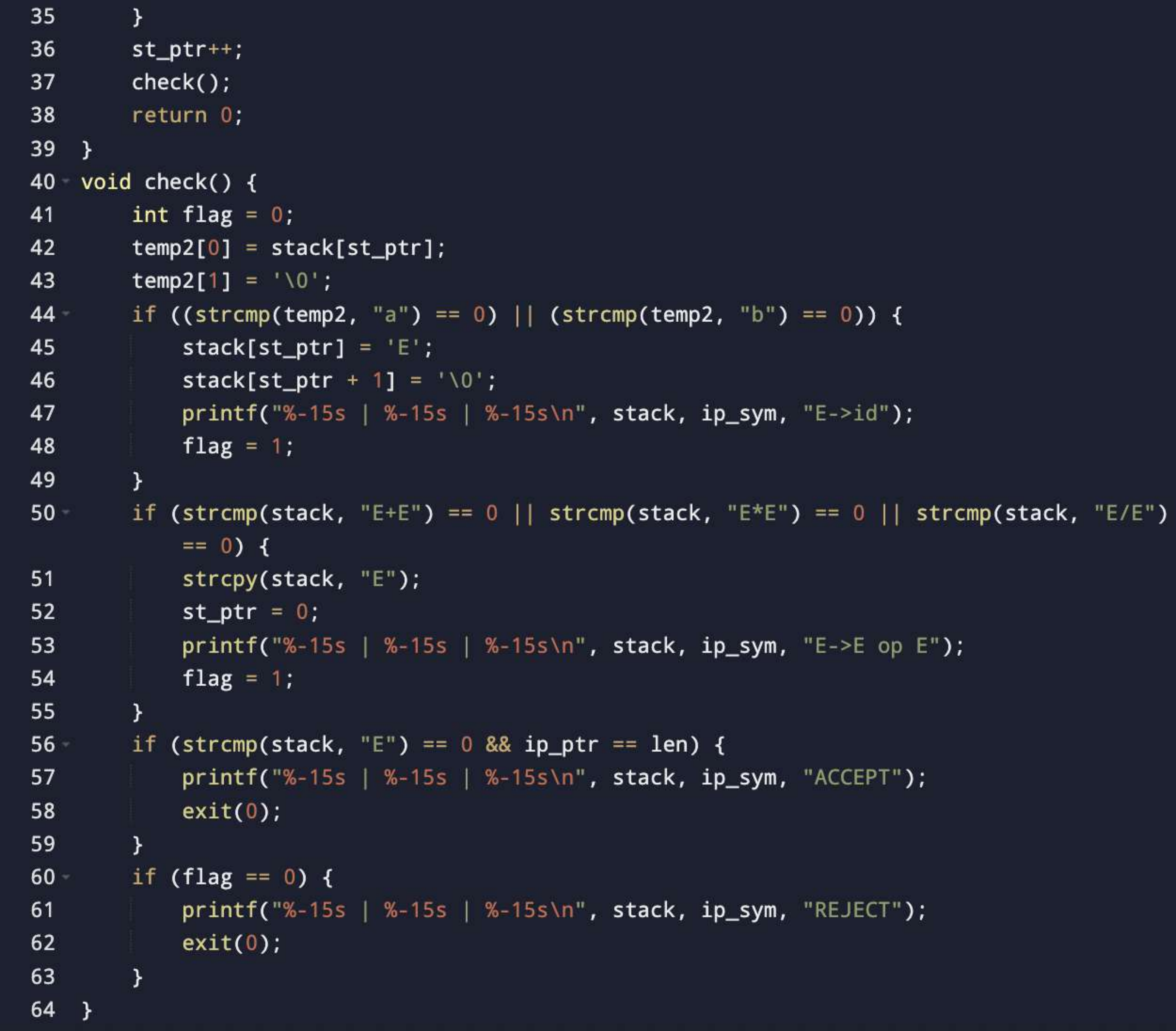
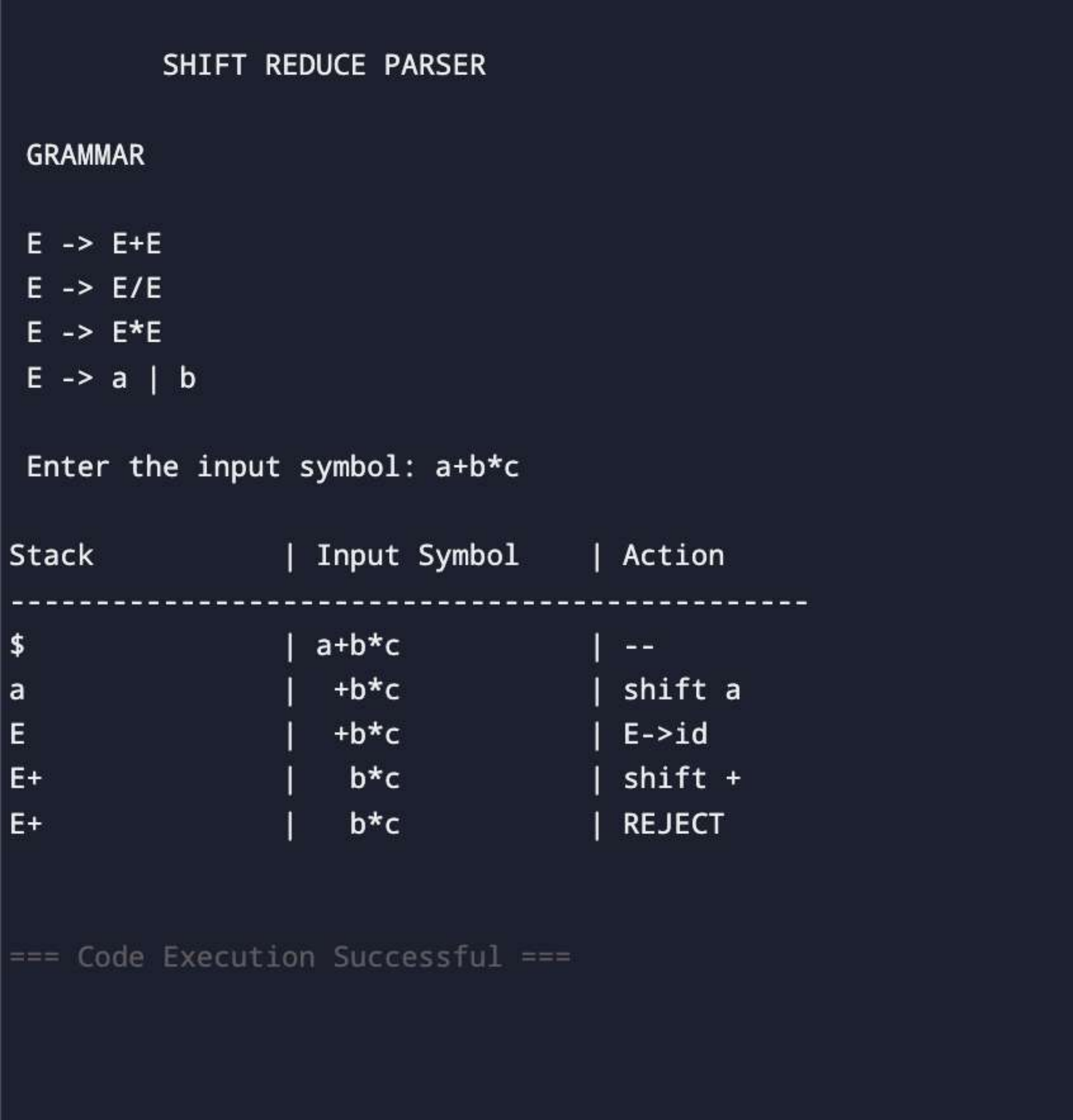


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
char ip_sym[50], stack[50];
int ip_ptr = 0, st_ptr = 0, len;
char temp[2], temp2[2], act[20];
void check();
int main() {
    printf("\n\t\t SHIFT REDUCE PARSER\n");
    printf("\n GRAMMAR\n");
    printf("\n E -> E+E\n E -> E/E\n E -> E*E\n E -> a | b\n");
    printf("\n Enter the input symbol: ");
    scanf("%s", ip_sym);
    printf("\n%-15s | %-15s | %-15s\n", "Stack", "Input Symbol", "Action");
    printf("----------------------------------------------\n");
    printf("%-15s | %-15s | %-15s\n", "$", ip_sym, "--");
    strcpy(act, "shift ");
    temp[0] = ip_sym[ip_ptr];
    temp[1] = '\0';
    strcat(act, temp);
    len = strlen(ip_sym);
    for (int i = 0; i < len; i++) {
        stack[st_ptr] = ip_sym[ip_ptr];
        stack[st_ptr + 1] = '\0';
        ip_sym[ip_ptr] = ' ';
        ip_ptr++;
        printf("%-15s | %-15s | %-15s\n", stack, ip_sym, act);
        strcpy(act, "shift ");
        temp[0] = ip_sym[ip_ptr];
        temp[1] = '\0';
        strcat(act, temp);
        check();
        st_ptr++;
```



```c
    }
    st_ptr++;
    check();
    return 0;
}
void check() {
    int flag = 0;
    temp2[0] = stack[st_ptr];
    temp2[1] = '\0';
    if ((strcmp(temp2, "a") == 0) || (strcmp(temp2, "b") == 0)) {
        stack[st_ptr] = 'E';
        stack[st_ptr + 1] = '\0';
        printf("%-15s | %-15s | %-15s\n", stack, ip_sym, "E->id");
        flag = 1;
    }
    if (strcmp(stack, "E+E") == 0 || strcmp(stack, "E*E") == 0 || strcmp(stack, "E/E") == 0) {
        strcpy(stack, "E");
        st_ptr = 0;
        printf("%-15s | %-15s | %-15s\n", stack, ip_sym, "E->E op E");
        flag = 1;
    }
    if (strcmp(stack, "E") == 0 && ip_ptr == len) {
        printf("%-15s | %-15s | %-15s\n", stack, ip_sym, "ACCEPT");
        exit(0);
    }
    if (flag == 0) {
        printf("%-15s | %-15s | %-15s\n", stack, ip_sym, "REJECT");
        exit(0);
    }
}
```



```
         SHIFT REDUCE PARSER

 GRAMMAR

 E -> E+E
 E -> E/E
 E -> E*E
 E -> a | b

 Enter the input symbol: a+b*c

Stack            | Input Symbol    | Action
-----------------------------------------------
$                | a+b*c           | --
a                |  +b*c           | shift a
E                |  +b*c           | E->id
E+               |   b*c           | shift +
E+               |   b*c           | REJECT


=== Code Execution Successful ===
```