shiftreduceparser.c

```c
#include <stdio.h>
#include <string.h>

int k = 0, i = 0, j = 0, c = 0;
char a[16], stk[15], act[10] = "SHIFT->";
void check();

int main() {
    printf("GRAMMAR is:\nE -> E+E\nE -> E*E\nE -> (E)\nE -> id\n");
    printf("Enter input string: ");
    scanf("%s", a);
    c = strlen(a);

    printf("stack\tinput\taction\n");
    for (k = 0; j < c; k++, i++, j++) {
        if (a[j] == 'i' && a[j + 1] == 'd') {
            strncpy(&stk[i], "id", 2);
            stk[i + 2] = '\0';
            a[j] = a[j + 1] = ' ';
            printf("$%s\t%s$\t%sid\n", stk, a, act);
            check();
            j++;
        } else {
            stk[i] = a[j];
            stk[i + 1] = '\0';
            a[j] = ' ';
            printf("$%s\t%s$\t%ssymbol\n", stk, a, act);
            check();
        }
    }
    return 0;
}

void check() {
    const char *reduceMsg = "REDUCE TO E";
    int z;

    for (z = 0; z < c; z++) {
        if (strncmp(&stk[z], "id", 2) == 0) {
            stk[z] = 'E';
            stk[z + 1] = '\0';
            printf("$%s\t%s$\t%s\n", stk, a, reduceMsg);
            break;
        }
    }

    for (z = 0; z < c - 2; z++) {
        if ((stk[z] == 'E' && stk[z + 1] == '+' && stk[z + 2] == 'E') ||
            (stk[z] == 'E' && stk[z + 1] == '*' && stk[z + 2] == 'E')) {
            stk[z] = 'E';
            stk[z + 1] = stk[z + 2] = '\0';
```

```c
                printf("$%s\t%s$\t%s\n", stk, a, reduceMsg);
                i -= 2;
                break;
            }
        }

        for (z = 0; z < c - 2; z++) {
            if (stk[z] == '(' && stk[z + 1] == 'E' && stk[z + 2] == ')') {
                stk[z] = 'E';
                stk[z + 1] = stk[z + 2] = '\0';
                printf("$%s\t%s$\t%s\n", stk, a, reduceMsg);
                i -= 2;
                break;
            }
        }
}

/*
ALGORITHM
ALGORITHM:
1. Start
2. Initialize all necessary variables and character arrays
3. Display the defined grammar rules for arithmetic expression
4. Get input from user
5. Parsing process, loop through each characters in input string:
    5.1 if character is 'id':
        5.1.1 shift it into stack
        5.1.2 print action as 'SHIFT -> id'
    5.2 if it is other symbols:
        5.2.1 push them onto stack
        5.2.2 print action as 'SHIFT -> symbols'
    5.3 after each operation, call the check() function
6. check() function:
    6.1 reduce 'id' to 'E'
    6.2 perform reduction based on grammar rules
    6.3 after each reduction, print the updated stack and reduction action as 'REDUCE to E'
7. Stop
*/
```