//Ankit Kumar 2K20/CO/72 InfixToPostfix

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 1000

typedef struct OperatorStack stack;

struct OperatorStack {

    char \*op;

    int top;

    int size;

};

int isEmpty(stack \*st) { return st->top == -1; }

int isFull(stack \*st) { return st->top == st->size - 1; }

char pop(stack \*st) {

    if (isEmpty(st)) return 0;

    char ch = st->op[st->top];

    st->top--;

    return ch;

}

void push(stack \*st, char ch) {

    if (isFull(st)) return;

    st->top++;

    st->op[st->top] = ch;

}

int isp(char ch) {

    switch (ch) {

    case '+': return 1;

    case '-': return 1;

    case '\*': return 2;

    case '/': return 2;

    case '%': return 2;

    case '^': return 3;

    case '(': return 0;

    }

    return -1;

}

int icp(char ch) {

    switch (ch) {

    case '+': return 1;

    case '-': return 1;

    case '\*': return 2;

    case '/': return 2;

    case '%': return 2;

    case '^': return 4;

    case '(': return 4;

    }

    return -1;

}

int operand(char ch) { return ch >= 'a' && ch <= 'z'; }

int operator(char ch) { return ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '^' || ch == '(' || ch == '%'; }

int errorOn(char infix[]) {

    int i = 0, operators = 0, operands = 0;

    while (infix[i] != '$') {

        if (operand(infix[i])) operands++;

        else if (operator(infix[i]) && !(infix[i] == '(' || infix[i] == ')'))

            operators++;

        if (!(operands == (operators + operands + 1) / 2 && operators == (operators + operands) / 2)) return i;

        i++;

    }

    return -1;

}

void infixToPostfix(char infix[]) {

    char postfix[MAX\_SIZE];

    stack \*st = (struct OperatorStack \*)malloc(sizeof(stack));

    st->op = (char \*)malloc(MAX\_SIZE \* sizeof(char));

    st->top = -1;

    st->size = MAX\_SIZE;

    int i = 0, j = 0;

    while (infix[i] != '$') {

        if (operand(infix[i])) postfix[j++] = infix[i];

        else if (infix[i] == ')') {

            while (st->op[st->top] != '(')

                postfix[j++] = pop(st);

            pop(st);

        } else if (operator(infix[i])) {

            while (icp(infix[i]) <= isp(st->op[st->top]))

                postfix[j++] = pop(st);

            push(st, infix[i]);

        }

        i++;

    }

    while (st->top >= 0)

        postfix[j++] = pop(st);

    postfix[j++] = '$';

    postfix[j] = '\0';

    printf("Postfix Expression: %s\n\n", postfix);

}

int main() {

    char infix[MAX\_SIZE];

    printf("Enter Infix Expression followed by '$' : ");

    gets(infix);

    int e = errorOn(infix);

    if (e == -1) infixToPostfix(infix);

    else

        printf("ERROR!! Check Infix char %d!!", e);

    return 0;

}

