//Ankit Kumar 2K20/CO/72 PostfixEvaluation

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 1000

typedef struct OperandStack stack;

struct OperandStack {

    float \*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;

    float f = st->op[st->top];

    st->top--;

    return f;

}

void push(stack \*st, float f) {

    if (isFull(st)) return;

    st->top++;

    st->op[st->top] = f;

}

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

int operator(char ch) { return 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])) {

            operators++;

            if (operators + 1 > operands) return i;

        }

        i++;

    }

    return operators + 1 == operands ? -1 : -2;

}

float operate(float a, float b, char ch) {

    switch (ch) {

    case '+': return a + b;

    case '-': return a - b;

    case '\*': return a \* b;

    case '/': return a / b;

    case '%': return (int)a % (int)b;

    case '^': return pow(a, b);

    }

    return INT\_MIN;

}

void evalPostfix(char postfix[]) {

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

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

    st->top = -1;

    st->size = MAX\_SIZE;

    int i = 0;

    printf("Randomly selecting values between 1 to 20 for variables\n");

    while (postfix[i] != '$') {

        if (operand(postfix[i])) {

            //Randomly selecting values between 1 to 20 for variables

            srand(i);

            float a = rand();

            a = (int)a % 20 + 1;

            printf("%c :%f; ", postfix[i], a);

            push(st, a);

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

            float a = pop(st), b = pop(st);

            push(st, operate(b, a, postfix[i]));

        }

        i++;

    }

    printf("\n%s = %f\n\n", postfix, st->op[st->top]);

}

int main() {

    char postfix[MAX\_SIZE];

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

    gets(postfix);

    int e = errorOn(postfix);

    if (e == -1) evalPostfix(postfix);

    else if (e == -2)

        printf("ERROR!! Too few Operators!");

    else

        printf("ERROR!! Check Postfix Char %d!!", e);

    return 0;

}

