Задача за Стек и Опашка

Трябва да симулирате работа на система, която получава от потребители изрази и ги изчислява. Работата на машината се управлява от получен отделно управляващ низ, който се състои от две команди: *read* и *calc*. Всяка от командите се кодира с един символ, съответно *R* и *C*. При получаване на команда *R* трябва да се прочете един израз от стандартния вход и да се съхрани в опашка от чакащи обработка низове, а при команда *C* да се изчисли (ако може) един израз от събраните в опашката с прочетени. Управляващият низ се получава като аргумент от командния ред.

Изразите са във вида:

<var> = <expression>

и задават стойността на точно една променлива с име *<var>* да бъде стойността на *<expression>*.

<expression> е аритметичен израз който може да съдържа цели числа, дробни числа с разделител между цяла и дробна част символа точка, операции, променливи и функции. Допустими са произволен брой празни позиции между елементите на израза.

Променливите са символи от латинската азбука, малки или големи (не се прави разлика между малки и големи), и с тях се свързват определени стойности. Тези стойности се получават като резултат от изчислението на друг израз.

Операциите могат да са унарни (+ и -, със съответната им семантика) и бинарни (+, -, *, /, ^, със стандартната семантика за дробни числа, като ^ означава степенуване). Бинарните операции +, -, * и / са с лява асоциативност, а ^ е с дясна. При изчисление на даден алгебричен израз се спазва обичайният приоритет на операциите (най-приоритетни са унарните, след това степенуване, след това умножение и деление и най-ниско приоритетни са събиране и изваждане). Допълнително даден алгебричен израз може да съдържа и скоби (), които променят приоритета по стандартния начин.

Функциите представляват специални подизрази. Всяка функция има уникално име и фиксиран брой аргументи. Аргументите на функцията се ограждат в скоби непосредствено след името ѝ и се разделят с точка-запетая ‘;’. Аргументи могат да са произволни изрази. Функциите могат да имат произволен брой аргументи, включително 0.
Вашата програма трябва да поддържа следните функции:

MIN(a_1 ; ... ; a_n) -> изчислява минималното от стойностите на a_1 до a_n
MAX(a_1 ; ... ; a_n) -> изчислява максималното от стойностите на a_1 до a_n.
AVG(a_1; ... ; a_n) -> изчислява средноаритметичното от стойностите на изразиазите a_1 до a_n.
SUM(a_1; ... ; a_n) -> изчислява сумата на стойностите на изразиазите a_1 до a_n.
PRD(a_1; ... ; a_n) -> изчислява произведенията на стойностите на изразиазите a_1 до a_n.

Обърнете внимание, че аргумент на функция може да бъде и променлива, както и друг алгебричен израз. Имената на функциите могат да бъдат изписани в израза с малки или главни букви (не трябва да се прави разлика между двете).

Примери за изрази са:

1 + 3 * 7 + 9 (със стойност 31)
13 + SUM(1 ; 3 ; 4+3 ; 9) (със стойност 33)

Когато в израз участва променлива, то има две възможности - ако тази променлива вече е получила стойност, то тя се използва в изчислението. Ако няма стойност, то изчислението на израза се отлага - изразът се изпраща обратно в края на опашката, където ще изчака да бъде изчислен по-късно. Важно изискване е, да се запази вече извършената работа. Например ако изразът е **`s = 3 + 5*7 + (3*5) * x`** и нямаме стойност за ***x***, то би трябвало в опашката да се сложи израз, който е частично обработен (имате свобода да прецените как точно да постигнете това) - например да е сведен до еквивалент на **`s = 38+15*x`**. Съветваме ви, да не съхранявате изразите в текстов вид, а да използвате подходящ клас. Възможно е израз да съдържа променливата, която сам дефинира. Тогава нейната стойност трябва да бъде получена от друг израз.

След като израз бъде успешно изчислен, то неговата стойност се асоциира със съответната променлива и тази стойност може да бъде използвана в изчислението на всеки следващ израз. Ако съответната променлива вече има стойност, то тя се заменя с новата. Всяко изчисление трябва да изведе на екрана или стойността на израза или съобщение, че изчислението е отложено.

Пример за работа на системата, която получава следния контролен низ:
RRRRCCCCRCC

А потребителят въвежда следните редове:
```
x = 2 * SUM(x; 4.5+2*y; 1.5)
y = 5+6 * -2
z = SUM(MIN(1; 2; 3); SUM(1))
x = 5*6
y = 2 * x
```

Очаква се следния изход:
```
Expression (1) was read.
Expression (2) was read.
Expression (3) was read.
Expression (4) was read.
Expression (1) was postponed.
Expression (2) was calculated. The value of y is now -7.
Expression (3) was calculated. The value of z is now 2.
Expression (4) was calculated. The value of x is now 30.
Expression (5) was read.
Expression (1) was calculated. The value of x is now 44.
Expression (5) was calculated. The value of y is now 88.
```

Ако в края на контролният израз останат неизчислени изрази, то програмата трябва да попита потребителя дали да ги изчисли (ако е възможно) или да ги изхвърли и да приключи работа. Трябва да съобщавате с подходящи съобщения за всякакви грешки по време на работа, но да продължите работата на машината доколкото това е възможно. Например да съобщите за невалидни символи във контролния низ, за команда *`calc`* в случай, че опашката е празна, за синтактични и семантични грешки в изразите и така нататък. При подобни грешки прескачате грешните команди или изрази и опитвате да продължите работа.