Southeastern European Regional Programming Contest
Bucharest, Romania
October 15, 2011

Problem I
Term Generator

Input File: I.IN
Output File: standard output
Program Source File: I.C, I.CPP, I.JAVA

A formula has the syntax shown in figure 1(a). If a formula has the particular syntax given in figure 1(b) we say that the formula is in Normal Form (NF).

```
<formula>::= <variable> |
             (+<formulae>) | (*<formulae>)
<variable>::= a lower case letter from the English
              alphabet
<formulae>::= <formula> |
              <formula><formulae>
```

a) The general syntax of a formula

```
<NF_formula>::= <term> | (+<terms>)
<term>::= <variable> | (*<variables>)
<terms>::= <term><term> | <term><terms>
<variables>::= <variable><variable> |
               <variable><variables>
```

b) The syntax of a NF formula

Figure 1. Formula syntax

A formula is converted to NF according to the rewriting rules given below, where **F** represents a formula, **S** stands for a non empty sequence of formulae, and **s** and **s'** denote possibly empty sequences of formulae. Applying a rewriting rule **q**→**r** on a formula **F** means to substitute by **r** a part of **F** that matches the pattern **q**, as shown in figure 2. The conversion terminates when no rewriting rule can be applied. The conversion terminates for any formula, and the result is unique regardless which rules are applied on which parts of the formula and in which order.

```
1. (+F)→F
2. (*F)→F
3. (+s(+S)s')→(+sSs')
4. (*s(*S)s')→(*sSs')
5. (*s(+FS)s')→(+(*sFs')(*s(+S)s'))
```

```
(+(*(+(*ab)(+a))b))       -1->
(+(*(+(*ab)a)b))          -5->
(+(+(*(*ab)b)(*(+a) b)))  -1->
(+(+(*(*ab)b)(*ab)))      -4->
(+(+(*abb)(*ab)))         -1->
(+(*abb)(*ab))
```

Figure 2. Converting a formula to NF

Let **NF(F)** be the normal form of the formula **F**. The problem is to write a term generator that for a given formula **F** and a number **k** outputs the next **k** terms from **NF(F)** in the order in which they appear in **NF(F)**. If the terms are exhausted, the generator continues from the first term of **NF(F)**. For example, consider **F=(+(*(+(*ab)(+a))b))**, and **NF(F)=(+(*abb)(*ab))** as in figure 2. Generating the first term from **NF(F)** yields **(*abb)**. Generating two more terms produces **(*ab)**, **(*abb)**. Notice that if **NF(F)** contains similar terms, as in the last example in figure 3, these terms are considered distinct.

Write a term generator that reads sets of data from the standard input. The content of a data set is **F** k_1...k_n 0, n>0, where **F** is a formula, and k_1...k_n are long integers different than 0. For each k_i, i=1,n, the program generates the next $|k_i|$ terms from **NF(F)** and, if k_i>0, prints these terms on the standard output. Each printed term starts from the beginning of a line and there are no white spaces between the characters of the term. The generated terms are not printed if k_i<0. White spaces are used freely in the input. Each formula **F** in the input has at most 150 characters and each term of **NF(F)** is at most 80 characters long, not counting white spaces. The input data terminate with an end of file, and are correct.

Input	Output
(+(*(+(*ab)(+a))b))	(*ab)
-3 1 1 0	(*abb)
(+xyy) -2 1 0	y

Figure 3. Input/output sample