# Expression Trees

An expression with binary operators * and + is defined recursively as

1. A single variable is an expression.

2. If $E_1$ and $E_2$ are expressions then so are $E_1 + E_2$ and $E_1 * E_2$.

An expression can be considered to be a labeled binary tree, where a variable corresponds to an empty tree and each node is labeled by either + or $*$. Such binary trees are called expression trees. In this problem, + and $*$ are assumed to be arbitrary operations that may not be associative or commutative.

Given the values for the variables in the expression, the expression can be evaluated by applying the operators + and $*$ to the values. The number of operations performed is then equal to the number of nodes in the expression tree.

Suppose you have a machine in which any number of $*$, or any number of + operations can be performed in a single step, but the two operations cannot be mixed in one step. In other words, any expression of the type $(v_1 * v_2) * (v_3 * (v_4 * v_5))$ can be evaluated in a single step, but $((v_1 * v_2) + v_3)$ requires 2 steps. You have to find the minimum number of steps on such a machine that are needed to evaluate the given expression. Suppose the machine has an arbitrary number of processors so that any number of steps can be executed in parallel, where each step is any expression with only $*$ or only + operations. A step can be executed only after all the operands in the expression have been evaluated. Assuming that each step takes unit time, and any number of steps can be executed in parallel, you have to find the minimum time required to evaluate the expression.

For the second part of the problem, you have to do the same thing except that there is a limit $k$ on the number of operands in an expression that can be evaluated in a single step. Thus the expression that can be evaluated in a single step has at most $k - 1$ $*$ or $k - 1$ + operations. Note that the first part is a special case with $k = n + 1$ where $n$ is the number of nodes in the tree. You will get half credit if you do the first part correctly.
**Input/Output:** The expression will be specified in postfix form, which is obtained by a postorder traversal of the expression tree. (See the slides or any other reference to see what this means.) An empty tree, which corresponds to a variable, will be denoted by the character '`e`' and the operators by '`*`' and '`+`'. The first line of input will contain the expression in postfix form. The next line will contain $t$, the number of test cases, where each test case will specify different values for the bound $k$ in the next $t$ lines. Note that the length of the string $l = 2n + 1$, where $n$ is the number of nodes in the expression tree. The product $l * t$ will be at most $10^7$ and $k$ will be at most $(l + 1)/2$, and at least 2, in each case. Each line of output should contain two numbers, the minimum number of steps required, and the minimum time required for each case, separated by a space. The first line should be the output for the first part without any bound. The subsequent lines should contain the outputs for each test case in the same order.

```
Sample Input                    Sample Output.
ee*ee**ee*+ee++                 3  2
2                               7  4
2                               5  3
3
```

**Note:** Although it is possible to solve the problem by just traversing the string once and using a stack, it may be better, especially for the second part, if the stack is used once to construct the expression tree, and the problem solved recursively using the tree. While this may be slightly less efficient, it is easier to understand and code. If the height of the tree is large, you may also need to set the stack size limit to unlimited.
**Submission:** Submit a single file named RollNo_8.cpp .
**Homework:** What happens if there is a bound $m$ on the number of processors, that is the number of steps that can be done in parallel?