The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem C
Flow Graph Complexity
Time Limit: 1 Second

Soohwan is leading the International Committee for Program Complexity (ICPC), and the major task of ICPC is measuring the complexity of program codes. A well-known representation of a program is the control flow graph, where the nodes represent the program constructs and the edges the possible flow of controls. The cyclomatic complexity is a popular way to measure the complexity of a flow graph. For the digraph $G(V, E)$ representing a flow graph, the cyclomatic complexity $C(G)$ is defined by the formula $C(G) = |E| - |V| + 2$. For instance, for the flow graphs G_1 and G_2 in Figure C.1, the complexities of them are computed to be 3 ($C(G_1) = C(G_2) = 9 - 8 + 2 = 3$).

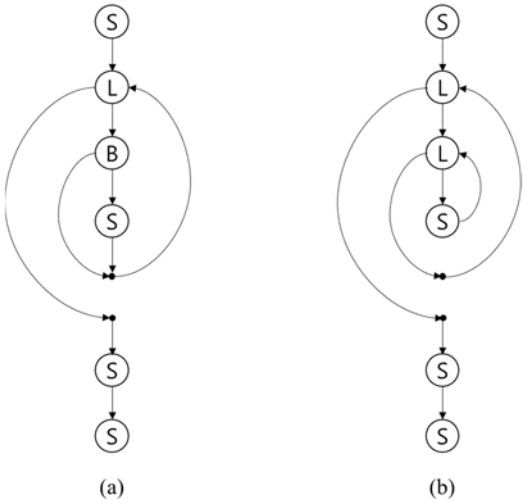

Figure C.1: Two flow graphs G_1 (a) and G_2 (b)

In Figure C.1, the labels B, L, and S denote the types of nodes: B for a branch, L for a loop, and S for a simple statement. The branch node B introduces additional forward edge skipping one or more statements. The loop node L, the target of an incoming backward edge, also introduces a forward edge exiting the loop. Note that the dots representing the closing points of loops or branches are nodes, too.

One critique for the cyclomatic complexity is that the backward edges introduced by loops are weighted equal to the forward edges. Therefore Soohwan devised a new measure imposing more weights on the backward edges than the forward ones. He classified the edges into two disjoint categories E_F for the forward and E_B for the backward ($E = E_F \cup E_B$ and $E_F \cap E_B = \emptyset$), and defined a new measure C_M, namely the flow graph complexity, using the formula $C_M(G) = |E_F| + W \times |E_B| - |V| + 2$ for a constant weight W for backward edges. Soohwan wants to validate this new measure.

You are to help Soohwan by writing a program calculating the new complexity for a given flow graph. For simplicity, only pretest loops (say, `while` loops in C) and only one-way branches (say, `if` statements without `else` clauses) are assumed. With this assumption, a flow graph can be represented by a string of node types defined as follows:

L for a loop node,
B for a branch node, and
S for a simple statement node.

L and B should be followed by the string representing its sub-structure enclosed by a pair of parentheses. Beware that the closing points are implicitly represented by parentheses. The sequence of node representations are separated by commas. According to this representation, the string representations of the graphs in Figure C.1 are as follows:

G_1: S,L(B(S)),S,S
G_2: S,L(L(S)),S,S

With the weight $W = 5$, the new complexities for them are $C_M(G_1) = |E_F| + W \times |E_B| - |V| + 2 = 8 + 5 \times 1 - 8 + 2 = 7$ and $C_M(G_2) = |E_F| + W \times |E_B| - |V| + 2 = 7 + 5 \times 2 - 8 + 2 = 11$.

Write a program to compute C_M given the weight W for backward edges and the string representing the flow graph.

Input

Your program is to read from standard input. The input starts with a line containing an integer, W ($1 < W \leq 27$), where W is the weight for the backward edge. In the following line, the string representation P for the flow graph is given. The length of P is less than 70,000. The string P consists of uppercase alphabets and punctuation symbols. To make the representation easy to read, P may contain spaces; the pairs of brackets [and] can be used instead of the pairs of parentheses and they should be matched. P may contain other punctuation symbols such as colon (:), semicolon (;), period (.) erroneously, which your program should detect them as invalid symbols. The input string P is invalid if (1) the parentheses and the brackets are unmatched, (2) it contains punctuation symbols other than parentheses, brackets, and commas, (3) the punctuation symbols are added or omitted wrongly such as S,,S, SS, or B,(S), and (4) it contains unknown types of node other than S, L, or B.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain the flow graph complexity $C_M(G)$ for the given G represented by P. If input string P is invalid, print -1 instead.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
5 S,L(L(S)),S,S	11

Sample Input 2	Output for the Sample Input 2
3 L(S, L[B(S),S])	8

Sample Input 3	Output for the Sample Input 3
11 B(S, L((F(S), S], S(S))	-1

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem C
Flow Graph Complexity
Time Limit: 1 Second

수환은 프로그램 복잡도 국제 위원회(ICPC: International Committee for Program Complexity)를 이끌고 있는데 위원회의 주요 임무는 프로그램 코드의 복잡도를 측정하는 것이다. 프로그램 표현법으로 제어 흐름 그래프(control flow graph)가 잘 알려져 있는데, 프로그램 구성 요소를 정점(node)으로 나타내고 가능한 제어 흐름을 간선(edge)으로 나타내는 방법이다. 순환복잡도(cyclomatic complexity)는 흐름 그래프의 복잡도 측정 방법으로 유명하다. 흐름 그래프를 나타내는 방향 그래프 $G(V,E)$에 대한 순환 복잡도 $C(G)$의 정의는 공식 $C(G) = |E| - |V| + 2$로 주어진다. 예컨대 그림 C.1의 그래프 G_1과 G_2에 대한 순환 복잡도는 3이다($C(G_1) = C(G_2) = 9 - 8 + 2 = 3$).

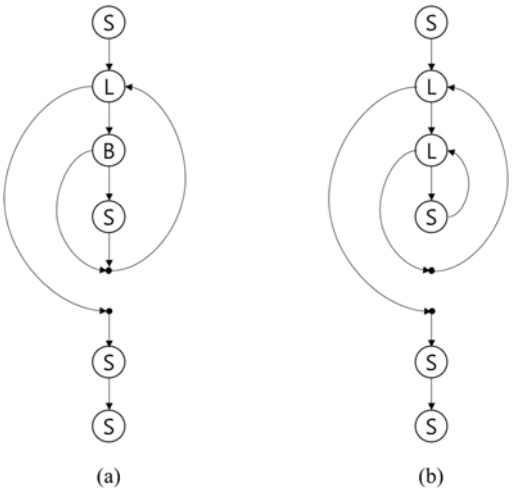

Figure C.1: 두 개의 흐름 그래프 (a) G_1과 (b) G_2

그림 C.1에서 레이블 B, L, S는 노드 종류를 나타내는데 B는 분기(branch), L은 반복(loop), S는 단순 문장을 나타낸다. 분기 노드 B에서는 하나 이상의 문장을 건너뛰는 순방향 간선(forward edge)이 하나 더 생성된다. 역방향 간선(backward edge)이 유입되는 반복 노드 L에서도 루프를 탈출하는 순방향 간선이 하나 더 생성된다. 반복과 분기가 끝나는 곳을 나타내는 점도 정점이라는 것에 주의하자.

순환 복잡도는 순방향 간선과 역방향 간선을 같은 비중으로 다룬다는 점에서 비판받고 있다. 그래서 수환은 순방향 간선보다 역방향 간선에 더 큰 가중치를 두는 새로운 복잡도를 고안하였다. 수환은 간선을 순방향 간선 E_F과 역방향 간선 E_B, 두 부류로 나누고($E = E_F \cup E_B$이고 $E_F \cap E_B = \emptyset$), 이른바 흐름 그래프 복잡도라는 새로운 측정 기준 C_M을 정의하였는데 이는 가중치 상수 W에 대하여 공식 $C_M(G) = |E_F| + W \times |E_B| - |V| + 2$으로 정의된다. 수환은 이 새로운 측정법의 타당성을 입증하려고 한다.

여러분은 주어진 흐름 그래프에 대해 새로운 복잡도를 계산하는 프로그램을 작성하여 수환을 도와주고자 한다. 문제를 간단히 하기 위해 조건을 먼저 검사하는 반복문(예컨대 C의 `while` 문)과 단방향 분

기문(예컨대 `else` 가 없는 `if` 문)만 허용된다고 가정하자. 이렇게 가정하면 흐름 그래프는 다음과 같은 노드 종류의 문자열로 나타낼 수 있다.

L: 반복문 노드
B: 분기문 노드
S: 단순 문장 노드

L과 B 다음에는 하부 구조를 나타내는 문자열을 괄호로 묶어 나타낸다. 반복문과 분기문의 끝을 나타내는 정점은 괄호를 통해 암묵적으로 표현된다는 점에 주의하자. 정점을 나타내는 문자열은 쉼표로 구분하여 나타낸다. 이 표기법에 따라 그림 C.1의 그래프를 문자열로 나타내면 다음과 같다.

G_1: S,L(B(S)),S,S
G_2: S,L(L(S)),S,S

가중치가 $W = 5$일 때 이들 그래프에 대한 새로운 복잡도는 $C_M(G_1) = |E_F| + W \times |E_B| - |V| + 2 = 8 + 5 \times 1 - 8 + 2 = 7$, $C_M(G_2) = |E_F| + W \times |E_B| - |V| + 2 = 7 + 5 \times 2 - 8 + 2 = 11$로 계산할 수 있다.

흐름 그래프 문자열과 역방향 간선 가중치 W가 주어졌을 때 C_M을 계산하는 프로그램을 작성하라.

입력
프로그램의 입력은 표준 입력으로 주어진다. 입력의 첫 줄에는 역방향 간선에 대한 가중치를 나타내는 정수 $W(1 < W \leq 27)$가 주어지고 그 다음 줄에는 흐름 그래프의 문자열 표기 P가 주어진다. P의 길이는 70,000이하이다. 문자열 P는 대문자 알파벳과 문장 부호로 구성된다. 읽기 편하도록 P에는 공백문자가 추가될 수 있고 괄호 대신 대괄호 [와]를 쓸 수 있는데 대괄호도 짝이 맞아야 한다. P에는 쌍점(:)이나 쌍반점(;), 마침표(.) 등이 실수로 포함될 수 있는데 프로그램은 이런 실수를 잘못된 입력으로 검출해야 한다. 입력 문자열 P는 다음과 같은 경우에 잘못된 입력인데, (1) 괄호나 대괄호가 맞지 않는 경우, (2) 괄호, 대괄호, 쉼표 외의 다른 문장 부호가 포함된 경우, (3) 문장 부호가 잘못 추가되거나 생략된 경우(예: S, , S나 SS, B, (S)), (4) S, L, B 외에 알려지지 않은 타입의 노드가 포함된 경우이다.

출력
프로그램의 출력은 표준 출력으로 출력한다. 입력에 대해 한 행만 출력하는데, 입력 문자열 P가 나타내는 그래프 G의 흐름 그래프 복잡도 $C_M(G)$를 출력한다. 문자열 P가 잘못되었다면 -1을 출력한다.

다음은 세 개의 입력에 대한 출력 값을 나타낸 예제이다.

입력 예제 1	**입력 예제 1 에 대한 출력**
5 S,L(L(S)),S,S	11

입력 예제 2	**입력 예제 2 에 대한 출력**
3 L(S, L[B(S),S])	8

입력 예제 3	**입력 예제 3 에 대한 출력**
11 B(S, L((F(S), S], S(S))	-1