# KCPC는 무엇의 약자일까? (acronym)

시간 제한: 1 초
메모리 제한: 512 MB

UCPC와는 다르게 KCPC는 무엇의 약자인지 이미 알려져있다. Korea University Collegiate Programming Contest가 바로 그것인데, 이미 KCPC가 무엇의 약자인지 알고 있는 상황에서 'KCPC는 무엇의 약어일까'를 내려고 한 박홍빈은 김이 팍 빠졌다. 하지만 박홍빈은 그럼에도 문제를 내고 싶었기에 문제를 약간 수정했다.

문자열의 **축약**이란 문자열에서 임의의 문자들을 제거하고 남은 문자들을 순서를 유지하며 이어붙여 새로운 문자열을 만드는 과정으로 예시는 다음과 같다.

- `"ABCDE"` → `"BD"`
- `"KoreaUniversityCollegiateProgrammingContest"` → `"KCPC"`

따옴표 "는 문자열의 경계를 표현하기 위한 것이지, 문자열의 일부가 아니다.

어떤 문자열 $S$를 축약해서 `"KCPC"`로 만들 수 있는지 확인하는 것은 이미 있는 문제이니 `"KCPC"`로도 주어진 문자열 $A$로도 축약 가능한지 확인해 보기로 하였다.

문제에서 요구하는 것은 문자열 $A$와 $S$가 주어졌을 때, 문자열 $S$를 축약해서 문자열 $A$를 만들 수 있는지, 또 문자열 $S$를 축약해서 `"KCPC"`로도 만들 수 있는지 확인하는 것이다.

대문자와 소문자는 다른 문자로 취급하며, 축약 과정은 독립적이다.

## 입력 형식

첫 번째 줄에 알파벳 대문자와 소문자로만 이루어진 문자열 $A$가 주어진다.

두 번째 줄에 알파벳 대문자와 소문자로만 이루어진 문자열 $S$가 주어진다.

각각의 문자열 길이는 최대 1000자이다.

## 출력 형식

첫 번째 줄에 문자열 $S$를 축약하여 문자열 $A$와 `"KCPC"`를 만들 수 있다면 `"KCPC!"`를 그렇지 않다면 `"KCPC?"`를 따옴표를 제외하고 출력한다.

## 예제

| 표준 입력 | 표준 출력 |
|---|---|
| LOVE<br>ILOVEKCPC | KCPC! |
| Cat<br>KcpCanDstRingS | KCPC? |
| CCCC<br>KCPCKCPC | KCPC! |

## 설명

첫 번째 예제의 경우 I<u>LOVE</u>KCPC와 ILOVE<u>KCPC</u>처럼 LOVE로도 KCPC로도 축약할 수 있다.

두 번째 예제의 경우 Kcp<u>C</u>an<u>D</u>st<u>R</u>ing<u>S</u>처럼 Cat으로 축약할 수는 있지만, KCPC로 축약할 수는 없다. 알파벳 대소문자를 구문함에 유의해야 한다.

세 번째 예제의 경우 K<u>C</u>P<u>C</u>K<u>C</u>P<u>C</u>와 <u>K</u>CP<u>C</u>K<u>C</u>P<u>C</u>처럼 CCCC로도 KCPC로도 축약할 수 있다.

## 참고 사항

입력받은 문자열의 길이를 각 언어별로 알아내어 출력하는 프로그램은 다음과 같다.

```c
// C에서 입력받은 단어의 길이 출력하기
#include <stdio.h>
#include <string.h>
char s[20];
int main() {
    scanf("%s", s);
    int slen = strlen(s);
    printf("%d\n", slen);
}
```

```cpp
// C++에서 입력받은 단어의 길이 출력하기
#include <iostream>
#include <string>
std::string s;
int main() {
    std::cin >> s; // std::string에 scanf는 직접적으로 사용할 수 없다
    int slen = int(s.length());
    std::cout << slen << '\n';
}
```

```python
# Python에서 입력받은 줄의 길이 알아내기
s = input()
slen = len(s)
print(slen)
```

```java
// Java에서 입력받은 단어의 길이 알아내기
import java.util.Scanner;
class StringLength {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String s = in.next();
        int slen = s.length();
        System.out.println(slen);
    }
}
```

언어별로 '단어'와 '줄'의 차이가 있지만, 이 문제에서는 단어와 줄이 같다고 보아도 된다.

# 초코칩 케이크 (cake)

시간 제한: 1 초
메모리 제한: 512 MB

2019년도 어언 한 달밖에 남지 않았다. 다사다난했던 2019년을 기념하기 위하여 상헌이는 고려대학교 프로그래밍 경진대회 운영진과 출제진과 나누어먹을 케이크를 장식하고자 한다.

균형과 일관성을 극도로 중시하는 상헌이는 정사각형 케이크를 샀고, 이를 가로 $n$줄 세로 $n$줄이 되게 조각으로 만들었다. 미니멀리즘이 듬뿍 담겨있는 이 케이크 위에는 아무 장식도 뿌려져있지 않다. 너무 밋밋하다고 생각한 상헌이는 케이크에 초코칩을 다음과 같이 여러 번 뿌려 장식을 하려고 한다.

- 한 가로줄을 선택하여, 이 가로줄에 속한 모든 조각에 초코칩을 1개 추가한다.
- 한 세로줄을 선택하여, 이 세로줄에 속한 모든 조각에 초코칩을 1개 추가한다.

상헌이는 가장 초코칩이 많이 뿌려져 있는 조각을 '가장 맛있는 조각'이라고 부른다.

매번 장식을 하는 상헌이는 가장 맛있는 조각의 개수를 신경쓰지 않을 수 없다. 다행히 상헌이는 프로그램을 작성하여 자신의 행동에 따른 가장 맛있는 조각의 개수를 성공적으로 계산해내었고, 이를 다음과 같이 문제로 만들었다. 케이크에 장식을 올리는 느낌으로 문제를 풀어보자.

## 입력 형식

첫 번째 줄에 두 정수 $n$, $q$가 공백으로 구분되어 주어진다. ($1 \le n \le 30,000, 1 \le q \le 100,000$)

$n$은 가로줄과 세로줄의 개수이며, $q$는 장식을 하는 횟수이다.

이후로 $q$개의 줄에 두 정수 $t$, $a$가 공백으로 구분되어 주어진다. ($1 \le t \le 2,\ 1 \le a \le n$)

$t$가 1이면 $a$번째 가로줄에, 2이면 $a$번째 세로줄에 있는 조각들에 초코칩을 하나씩 더한다.

## 출력 형식

각 줄마다 매 번 장식을 한 이후 그 상태의 가장 맛있는 조각의 개수를 출력한다.

모든 초코칩은 장식이 끝난 후에도 유지된다.

## 예제

| 표준 입력 | 표준 출력 |
|---|---|
| 3 2<br>1 1<br>1 3 | 3<br>6 |
| 1 3<br>1 1<br>2 1<br>1 1 | 1<br>1<br>1 |
| 4 5<br>1 1<br>1 4<br>2 3<br>1 4<br>2 2 | 4<br>8<br>2<br>1<br>2 |

## 설명

첫 번째 예제에 대한 설명은 다음과 같다.

- 첫 번째 장식은 1번째 가로줄에 초코칩을 뿌리는 것이므로, 3개의 조각에 초코칩이 1개, 6개의 조각에 초코칩이 0개 있어 가장 맛있는 조각은 3개이다.
- 두 번째 장식까지 하고 나면 6개의 조각에 초코칩이 3개, 나머지 3개의 조각에 초코칩이 0개 있어 가장 맛있는 조각은 6개이다.

두 번째 예제에선 조각이 1개밖에 없으므로, 장식 스타일과 상관없이 가장 맛있는 조각도 1개밖에 없다.

# 얼마나 예뻐? (pretty)

시간 제한: 1 초
메모리 제한: 512 MB

12월 25일은 크리스마스이다. 크리스마스를 맞은 피카츄는 지우를 위해 트리를 만들기로 했다. 금전적인 문제로 살아있는 나무를 구할 수 없게 된 피카츄는 나무 대신 그래프의 일종인 트리를 꾸미기로 했다.

**트리**란 모든 정점이 연결되어 있으며, 한 정점에서 다른 정점으로 가는 경로가 유일한 그래프이다. 트리에서 간선으로 연결된 두 정점 $u$와 $v$에 대해 $u$가 루트 정점에 더 가까운 정점이라고 할 때, $v$의 부모는 $u$이며 $u$는 $v$를 자식으로 가진다. 트리에 루트 정점은 단 하나 존재하며, 정의에 의해 루트 정점은 부모 정점이 없다. 정점 $x$의 **서브트리**란 $x$와 $x$의 자식들의 서브트리로 구성된 트리를 의미한다.

정점 $x$에서의 **전위 순회**는 트리를 특정한 규칙에 의해 방문하는 순서로, 다음과 같이 정의된다.

- 정점 $x$를 방문한다.
- 자식 정점의 번호의 오름차순으로 (작은 번호부터 큰 번호로) 전위 순회를 한다.

전위 순회의 정의에 의해 전위 순회는 정점별로 유일하다. 추가적으로, 트리의 전위 순회는 루트 정점에서의 전위 순회로 정의한다.

한 정점에서의 전위 순회는 그 서브트리의 모든 정점을 정확히 한 번만 방문하기 때문에 의미가 있다.

수학적 정의가 귀찮았지만 트리를 그리는 건 좋았던 피카츄는 인터넷 검색을 통해 1번 정점이 루트인 트리를 그렸다. 트리를 더 장식하기 위해 각 정점에 괄호(여는 괄호 '(' 또는 닫는 괄호 ')')를 한 개씩 그렸는데, 트리를 본 지우는 정점이 다음의 성질을 만족하면 아름답다고 정의했다.

- 정점 $x$에서의 전위 순회를 하며, 방문한 순서대로 각 정점에 부여된 괄호를 추가해가며 문자열을 얻을 수 있다. 이렇게 만들 수 있는 괄호 문자열이 **올바른 괄호 문자열**이 존재한다면 $x$는 아름답다.

올바른 괄호 문자열은 다음과 같이 정의된다.

1. 빈 문자열은 올바른 괄호 문자열이다.
2. A가 올바른 괄호 문자열이면 괄호를 씌운 (A)도 올바른 괄호 문자열이다.
3. A와 B가 올바른 괄호 문자열이면 둘을 이어붙인 AB도 올바른 괄호 문자열이다.

지우는 트리의 아름다움을 트리의 정점 중 아름다운 정점의 수라고 정의했다. 피카츄를 도와 피카츄가 꾸민 트리가 얼마나 아름다운지 구하자.

## 입력 형식

첫 번째 줄에 노드의 개수 $N(2 \le N \le 200,000)$이 주어진다.

두 번째 줄에는 0과 1로만 이루어진 $N$개의 정수 $a_1, a_2, \cdots, a_N$가 공백으로 구분되어 주어진다. $a_i$가 0이면 정점 $i$에 여는 괄호 '('가 적혀 있으며, 1이면 ')'이 적혀 있다.

이후 $N-1$개 줄에 걸쳐, 두 정수 $a$와 $b$가 공백으로 구분되어 주어진다. $(1 \le a, b \le N)$ 이는 정점 $a$와 정점 $b$가 간선으로 연결되어 있음을 의미한다.

입력으로 들어오는 그래프는 트리이다.

## 출력 형식

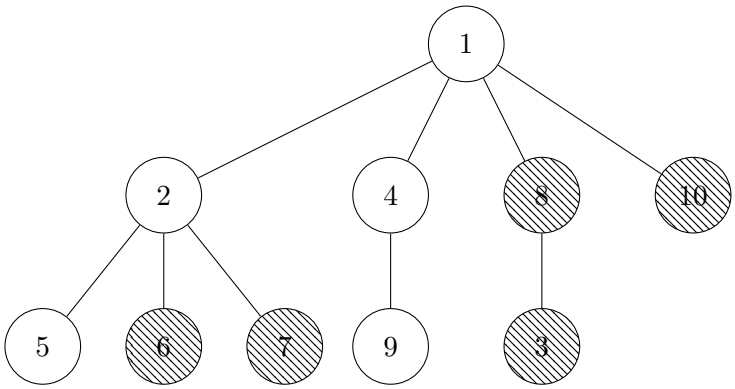
첫 번째 줄에 피카츄가 만든 트리의 아름다움을 출력한다.

## 예제

| 표준 입력 | 표준 출력 |
| --- | --- |
| 10<br>0 0 1 0 0 1 1 1 0 1<br>1 2<br>1 8<br>1 4<br>1 10<br>2 5<br>6 2<br>2 7<br>8 3<br>4 9 | 2 |

## 설명

첫 번째 예제에서 닫는 괄호가 있는 칸에 빗금을 쳐서 그리면 다음과 같다.




정점 1에서의 전위 순회는 $1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 3 \rightarrow 10$이고, 올바른 괄호 문자열 ((())(()))을 생성하기에 아름다운 정점이다.

정점 2에서의 전위 순회는 $2 \rightarrow 5 \rightarrow 6 \rightarrow 7$으로, 올바른 괄호 문자열 (())을 생성하기에 아름다운 정점이다.

그 외에 아름다운 정점은 존재하지 않는다.