Every day, Danny buys one sweet from the candy store and eats it. The store has m types of sweets, numbered from 1 to m. Danny knows that a balanced diet is important and is applying this concept to his sweet purchasing. To each sweet type i, he has assigned a *target fraction*, which is a real number f_i ($0 < f_i \leq 1$). He wants the fraction of sweets of type i among all sweets he has eaten to be roughly equal to f_i. To be more precise, let si denote the number of sweets of type i that Danny has eaten, and let $n = \Sigma_{i=1}^{m} s_i$. We say the set of sweets is *balanced* if for every i we have

$$nf_i - 1 < s_i < nf_i + 1.$$

Danny has been buying and eating sweets for a while and during this entire time the set of sweets has been balanced. He is now wondering how many more sweets he can buy while still fulfilling this condition. Given the target fractions f_i and the sequence of sweets he has eaten so far, determine how many more sweets he can buy and eat so that at any time the set of sweets is balanced.

Input

The input file contains several test cases, each of them as described below.

The input consists of three lines. The first line contains two integers m ($1 \leq m \leq 10^5$), which is the number of types of sweets, and k ($0 \leq k \leq 10^5$), which is the number of sweets Danny has already eaten.

The second line contains m positive integers $a_1, \ldots, a_m$. These numbers are proportional to $f_1, \ldots, f_m$, that is,

$$f_i = \frac{a_i}{\Sigma_{j=1}^{m} a_j}.$$

It is guaranteed that the sum of all a_j is no larger than 10^5.

The third line contains k integers $b_1, \ldots, b_k$ ($1 \leq b_i \leq m$), where each b_i denotes the type of sweet Danny bought and ate on the i-th day. It is guaranteed that every prefix of this sequence (including the whole sequence) is balanced.

Output

For each test case, on a line by itself, display the maximum number of additional sweets that Danny can buy and eat while keeping his diet continuously balanced. If there is no upper limit on the number of sweets, display the word '`forever`'.

Sample Input

```
6 5
2 1 6 3 5 3
1 2 5 3 5
6 4
2 1 6 3 5 3
1 2 5 3
```

Sample Output

```
1
forever
```