A. Saitama Destroys Hotel

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Saitama accidentally destroyed a hotel again. To repay the hotel company, Genos has volunteered to operate an elevator in one of its other hotels. The elevator is special — it starts on the top floor, can only move down, and has infinite capacity. Floors are numbered from 0 to s and elevator initially starts on floor s at time 0.

The elevator takes exactly 1 second to move down exactly 1 floor and negligible time to pick up passengers. Genos is given a list detailing when and on which floor passengers arrive. Please determine how long in seconds it will take Genos to bring all passengers to floor 0.

Input

The first line of input contains two integers n and s ($1 \le n \le 100$, $1 \le s \le 1000$) — the number of passengers and the number of the top floor respectively.

The next n lines each contain two space-separated integers f_i and t_i ($1 \le f_i \le s$, $1 \le t_i \le 1000$) — the floor and the time of arrival in seconds for the passenger number i.

Output

Print a single integer — the minimum amount of time in seconds needed to bring all the passengers to floor 0.

Examples

input

```
3 7
2 1
3 8
5 2
```

output

```
11
```

input

```
5 10
2 77
3 33
8 21
9 12
10 64
```

output

```
79
```

Note

In the first sample, it takes at least 11 seconds to bring all passengers to floor 0. Here is how this could be done:

1. Move to floor 5: takes 2 seconds.

2. Pick up passenger 3.

3. Move to floor 3: takes 2 seconds.

4. Wait for passenger 2 to arrive: takes 4 seconds.

5. Pick up passenger 2.

6. Go to floor 2: takes 1 second.

7. Pick up passenger 1.

8. Go to floor 0: takes 2 seconds.

This gives a total of $2 + 2 + 4 + 1 + 2 = 11$ seconds.

B. Hamming Distance Sum

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Genos needs your help. He was asked to solve the following programming problem by Saitama:

The length of some string s is denoted $|s|$. The Hamming distance between two strings s and t of equal length is defined as $\sum_{i=1}^{|s|} |s_i - t_i|$, where s_i is the i-th character of s and t_i is the i-th character of t. For example, the Hamming distance between string "`0011`" and string "`0110`" is $|0 - 0| + |0 - 1| + |1 - 1| + |1 - 0| = 0 + 1 + 0 + 1 = 2$.

Given two binary strings a and b, find the sum of the Hamming distances between a and all contiguous substrings of b of length $|a|$.

Input

The first line of the input contains binary string a ($1 \le |a| \le 200\,000$).

The second line of the input contains binary string b ($|a| \le |b| \le 200\,000$).

Both strings are guaranteed to consist of characters '`0`' and '`1`' only.

Output

Print a single integer — the sum of Hamming distances between a and all contiguous substrings of b of length $|a|$.

Examples

input
```
01
00111
```
output
```
3
```

input
```
0011
0110
```
output
```
2
```

Note

For the first sample case, there are four contiguous substrings of b of length $|a|$: "`00`", "`01`", "`11`", and "`11`". The distance between "`01`" and "`00`" is $|0 - 0| + |1 - 0| = 1$. The distance between "`01`" and "`01`" is $|0 - 0| + |1 - 1| = 0$. The distance between "`01`" and "`11`" is $|0 - 1| + |1 - 1| = 1$. Last distance counts twice, as there are two occurrences of string "`11`". The sum of these edit distances is $1 + 0 + 1 + 1 = 3$.

The second sample case is described in the statement.

C. Chain Reaction

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n beacons located at distinct positions on a number line. The i-th beacon has position a_i and power level b_i. When the i-th beacon is activated, it destroys all beacons to its left (direction of decreasing coordinates) within distance b_i inclusive. The beacon itself is not destroyed however. Saitama will activate the beacons one at a time from right to left. If a beacon is destroyed, it cannot be activated.

Saitama wants Genos to add a beacon **strictly to the right** of all the existing beacons, with any position and any power level, such that the least possible number of beacons are destroyed. Note that Genos's placement of the beacon means it will be the first beacon activated. Help Genos by finding the minimum number of beacons that could be destroyed.

Input

The first line of input contains a single integer n ($1 \le n \le 100\,000$) — the initial number of beacons.

The i-th of next n lines contains two integers a_i and b_i ($0 \le a_i \le 1\,000\,000$, $1 \le b_i \le 1\,000\,000$) — the position and power level of the i-th beacon respectively. No two beacons will have the same position, so $a_i \ne a_j$ if $i \ne j$.

Output

Print a single integer — the minimum number of beacons that could be destroyed if exactly one beacon is added.

Examples

input

```
4
1 9
3 1
6 1
7 4
```

output

```
1
```

input

```
7
1 1
2 1
3 1
4 1
5 1
6 1
7 1
```

output

```
3
```

Note

For the first sample case, the minimum number of beacons destroyed is 1. One way to achieve this is to place a beacon at position 9 with power level 2.

For the second sample case, the minimum number of beacons destroyed is 3. One way to achieve this is to place a beacon at position 1337 with power level 42.

D. Zuma

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Genos recently installed the game Zuma on his phone. In Zuma there exists a line of n gemstones, the i-th of which has color c_i. The goal of the game is to destroy all the gemstones in the line as quickly as possible.

In one second, Genos is able to choose exactly one continuous substring of colored gemstones that is a palindrome and remove it from the line. After the substring is removed, the remaining gemstones shift to form a solid line again. What is the minimum number of seconds needed to destroy the entire line?

Let us remind, that the string (or substring) is called *palindrome*, if it reads same backwards or forward. In our case this means the color of the first gemstone is equal to the color of the last one, the color of the second gemstone is equal to the color of the next to last and so on.

Input

The first line of input contains a single integer n ($1 \le n \le 500$) — the number of gemstones.

The second line contains n space-separated integers, the i-th of which is c_i ($1 \le c_i \le n$) — the color of the i-th gemstone in a line.

Output

Print a single integer — the minimum number of seconds needed to destroy the entire line.

Examples

input

```
3
1 2 1
```

output

```
1
```

input

```
3
1 2 3
```

output

```
3
```

input

```
7
1 4 4 2 3 2 1
```

output

```
2
```

Note

In the first sample, Genos can destroy the entire line in one second.

In the second sample, Genos can only destroy one gemstone at a time, so destroying three gemstones takes three seconds.

In the third sample, to achieve the optimal time of two seconds, destroy palindrome 4 4 first and then destroy palindrome 1 2 3 2 1.

E. Marbles

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In the spirit of the holidays, Saitama has given Genos two grid paths of length n (a weird gift even by Saitama's standards). A grid path is an ordered sequence of neighbouring squares in an infinite grid. Two squares are neighbouring if they share a side.

One example of a grid path is $(0, 0) \to (0, 1) \to (0, 2) \to (1, 2) \to (1, 1) \to (0, 1) \to (-1, 1)$. Note that squares in this sequence might be repeated, i.e. path has self intersections.

Movement within a grid path is restricted to adjacent squares within the sequence. That is, from the i-th square, one can **only move** to the $(i - 1)$-th or $(i + 1)$-th squares of this path. Note that there is only a single valid move from the first and last squares of a grid path. Also note, that even if there is some j-th square of the path that coincides with the i-th square, only moves to $(i - 1)$-th and $(i + 1)$-th squares are available. For example, from the second square in the above sequence, one can only move to either the first or third squares.

To ensure that movement is not ambiguous, the two grid paths will not have an alternating sequence of three squares. For example, a contiguous subsequence $(0, 0) \to (0, 1) \to (0, 0)$ **cannot occur** in a valid grid path.

One marble is placed on the first square of each grid path. Genos wants to get both marbles to the last square of each grid path. However, there is a catch. Whenever he moves one marble, the other marble will copy its movement if possible. For instance, if one marble moves east, then the other marble will *try* and move east as well. By *try*, we mean if moving east is a valid move, then the marble will move east.

Moving north increases the second coordinate by 1, while moving south decreases it by 1. Similarly, moving east increases first coordinate by 1, while moving west decreases it.

Given these two valid grid paths, Genos wants to know if it is possible to move both marbles to the ends of their respective paths. That is, if it is possible to move the marbles such that both marbles rest on the last square of their respective paths.

Input

The first line of the input contains a single integer n ($2 \le n \le 1\,000\,000$) — the length of the paths.

The second line of the input contains a string consisting of $n - 1$ characters (each of which is either 'N', 'E', 'S', or 'W') — the first grid path. The characters can be thought of as the sequence of moves needed to traverse the grid path. For example, the example path in the problem statement can be expressed by the string "NNESWW".

The third line of the input contains a string of $n - 1$ characters (each of which is either 'N', 'E', 'S', or 'W') — the second grid path.

Output

Print "YES" (without quotes) if it is possible for both marbles to be at the end position at the same time. Print "NO" (without quotes) otherwise. In both cases, the answer is case-insensitive.

Examples

input

```
7
NNESWW
SWSWSW
```

output

```
YES
```

input

```
3
NN
SS
```

output

```
NO
```

Note

In the first sample, the first grid path is the one described in the statement. Moreover, the following sequence of moves will get both marbles to the end: NNESWWSWSW.

In the second sample, no sequence of moves can get both marbles to the end.

Codeforces (c) Copyright 2010-2016 Mike Mirzayanov