A. Bear Girls

1 second, 256 megabytes

Limak and John are little polar bears. John is cute, and thus there are N bear girls who are interested in dating him.

Limak has decided to help John, so he went ahead and interviewed all N candidates. Limak is an excellent judge of bear girl beauty, so he was able to determine the exact beauty of each candidate. The beauties of the candidates turned out to be N distinct positive integers, with larger values representing more beautiful girls.

John has now planned a sequence of N dates: a single date with each of the girls. The order in which he is going to meet the girls was chosen at random, with each of the $N!$ permutations being equally likely.

Before going on any of the dates, John has two sources of information. First, Limak gave him an array of integers that lists the beauty of all the candidates. Additionally, John knows a fixed cost he has to pay for each date he actually goes to.

As soon as John starts going on the dates, he will be on his own, and Limak will not be able to assist him in any way. Immediately after each date, John will have to make a choice: either he rejects the current girl, or he accepts her. As soon as he accepts a girl, the dating is over: John will cancel all the remaining dates (and he will not pay the cost for those).

John's problem is that he is not such a good judge of beauty as Limak. John can only compare relative beauty. More precisely, whenever he goes on a date with a girl, he will be able to (always correctly) tell which of the previously dated girls were more beautiful and which ones were less beautiful than the current one.

John's primary goal is to accept exactly one girl. John's secondary goal is to maximize the expected value of $B - C$, where B is the beauty of the chosen girl and C is the total cost of the dates he went on (i.e., C is a fixed cost of a date times the number of dates). Find the optimal strategy for John and print the largest possible expected value of $B - C$.

Input

The first line contains two integers N and c — N ($1 \le N \le 10\,000$) is the number of candidates, and c ($0 \le c \le 10^9$) is the fixed cost John has to pay for each date.

The second line contains N distinct integers b_i, representing the beauties of the candidates ($1 \le b_i \le 10^9$; $b_i \ne b_j$ if $i \ne j$).

Output

Print a single number, representing the largest possible expected value of $B - C$. Your number must have an absolute or relative error smaller than 10^{-7}.

input
```
3 5
10 1 100
```
output
```
41.0000000000
```

input
```
3 1000
10 1 100
```
output
```
-963.0000000000
```

B. Bichrome Sky

3 seconds, 512 megabytes

There are n stars in the night sky. Each star randomly changes its color between red and blue. When you take a photo of the night sky, star i will be red with probability p_i and blue with probability $1 - p_i$. The colors of the stars in a photo are mutually independent random events.

You are going to take a photo of the night sky. The photo will be a two-dimensional plane. For each i, star i will be a point located at the coordinates (x_i, y_i).

Once you have the photo, you are going to draw two convex polygons: the convex hull of all red stars, and the convex hull of all blue stars. Note that these may sometimes be degenerate. It is even possible that all stars will have the same color, in which case the other convex hull is empty. Finally, you are going to compute the area of the intersection of those two polygons.

Given star coordinates and their color probabilities, find the expected area of the intersection.

Input

The first line contains a single integer n ($6 \le n \le 50$), the number of stars.

Each of the next n lines contains two integers and a decimal number with exactly three digits after the decimal point: x_i, y_i, and p_i — the coordinates of a star and the probability that it is red ($-1000 \le x_i, y_i \le 1000$; $0 \le p_i \le 1$).

All stars are located in distinct points. No three stars lie on the same line.

Output

Print a single floating-point number denoting the expected area of the intersection of the two convex hulls. Your return value must have a relative or absolute error less than 10^{-9}.

input
```
6
0 0 0.000
0 3 0.000
3 0 0.000
1 1 1.000
-2 1 1.000
1 -2 1.000
```
output
```
1.000000000
```

input
```
6
0 0 0.000
0 3 0.000
3 0 0.000
1 1 0.200
-2 1 0.300
1 -2 0.400
```
output
```
0.024000000
```

input
```
6
0 0 0.500
0 3 0.500
3 0 0.500
1 1 0.500
-2 1 0.500
1 -2 0.500
```
output
```
0.062500000
```

input

```
8
-1 3 0.249
-3 1 0.752
-2 -3 0.863
-3 -1 0.067
1 3 0.097
2 0 0.348
1 -3 0.526
0 -1 0.711
```

output

```
2.445160631
```

In the first example, stars 0, 1, and 2 are guaranteed to be blue. Stars 3, 4, and 5 are guaranteed to be red. The two convex hulls are uniquely determined. Their intersection is the square bounded by the lines $x = 0$, $x = 1$, $y = 0$, and $y = 1$. The area of this square is 1.

In the second example, stars 3, 4, and 5 are sometimes red and sometimes blue. If there are fewer than three red stars, the convex hull of all red stars will have no area, and therefore the area of the intersection will be 0. Thus, an intersection with a non-empty area only exists if all three of these stars are red. That happens with probability $0.2 \times 0.3 \times 0.4 = 0.024$. In that case, the area of the intersection is 1. Thus, the expected area of the intersection is $0.024 \times 1 = 0.024$.

C. Zandar's new game

2 seconds, 256 megabytes

Zandar has come up with a peculiar game to pass the time. Here are the rules:

You are provided with three arrays, p, q, and r, each comprising n non-negative integers. Your task is to determine the number of valid quadruples (x, y, z, w) that satisfy the following conditions:

- $1 \le x < y < z < w \le n$
- $g(p_x \text{ AND } p_y) = q_y$
- $g(p_y \text{ OR } p_z) = r_z$
- $g(p_z \text{ AND } p_w) = q_w$

Here, `AND` and `OR` signify bitwise operations.

$g(x)$ represents the count of 1 bits in the binary representation of x. For example, $g(6) = 2$ because the binary form of 6 is 110_2, which contains two 1 bits.

Input

The first line contains an integer, n, denoting the number of elements in the arrays ($1 \le n \le 10^5$).

Each of the following three lines contains the arrays p, q, and r, respectively ($0 \le p_i < 2^{16}$; $0 \le q_i, r_i \le 16$).

Output

Print the number of valid quadruples.

input

```
5
1 3 2 0 3
0 1 1 0 1
0 5 2 10 4
```

output

```
2
```

D. Deterministic Random Exploration

5 seconds, 512 megabytes

Jordan has a board game that includes a playing field, markers, and a specialized die. The playing field has n distinct positions, each with exactly one directed link leading to another position (potentially the same). The die has m faces numbered from 0 to $m-1$, and the probability of rolling the i-th face is p_i.

Jordan finds the game monotonous and decides to perform some numerical analysis. Initially, he places a marker randomly and uniformly on one of the positions. Subsequently, he rolls the die k times. If the die lands on s, Jordan then moves the marker through s transitions (remember, each position has exactly one transition).

Your task is to determine the probability that the marker will end up on each position after k rolling sequences.

All the probabilities in this problem are rational numbers reduced modulo $M = 998244353$. If a probability is $\frac{p}{q}$ in its simplest form with $q \bmod M \ne 0$, it should be expressed as $p \times q^{-1} \bmod M$.

Input

The first line contains three integers n, m, k — the number of positions, the number of die faces, and the number of rolls, respectively.

The second line contains n integers a_i ($1 \le a_i \le n$), where each a_i indicates the directed link from position i to position a_i.

The third line contains m integers p_i ($0 \le p_i < M$) — these represent the probabilities of each face of the die.

Constraints

$1 \le n \le 60000$

$4 \le m \le 100000$

$1 \le k \le 1000$

The sum of the p_i values modulo M equals 1.

Output

The output should consist of n lines. The i-th line should contain a single integer x_i ($0 \le x_i < M$), which is the probability that the marker will be at position i after k rolls.

input

```
4 5 1
3 3 2 4
748683265 748683265 748683265 748683265 0
```

output

```
935854081
686292993
623902721
748683265
```

In this example, the probabilities for faces 0, 1, 2, and 3 are $\frac{1}{4}$, and the probability for face 4 is zero.

The respective probabilities are $\frac{1}{16}$, $\frac{5}{16}$, $\frac{3}{8}$, $\frac{1}{4}$.

E. Tensioned Rope

2 seconds, 512 megabytes

On the plane, there exists an obstacle in the form of a simple polygon with vertices at points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The vertex m is connected to the vertex k with an elastic tensioned rope. Each point of the rope remains outside the polygon (but they can be on its boundary). The rope always tries to minimize its length since the friction force between the obstacle and the rope is zero. The ends of the rope are fixed at points (x_m, y_m) and (x_k, y_k), with no other points of the rope fixed.

As it happens, the shape of the rope is an arbitrary line that does not intersect or touch itself. What is the **maximum** possible length of this rope?

Input

The first line contains three integers n, m, k ($1 \le m, k \le n$, $m \ne k$).

Each of the next n lines contains two integers x_i, y_i — the vertices of the polygon in either clockwise or counterclockwise order. It is guaranteed that the described polygon is simple.

Constraints

$3 \le n \le 500$

$1 \le x_i, y_i \le 500$

Output

Output a single real number, representing the answer to the question. The answer will be considered correct if it has an **absolute** error of at most 10^{-6}.

input
```
4 2 4
100 100
200 100
200 200
100 200
```
output
```
200.0000000000
```

input
```
6 4 1
167 84
421 84
283 192
433 298
164 275
320 133
```
output
```
468.3361845326
```

F. Aerobics

10 seconds, 512 megabytes

The aerobics class begins. The trainer says, "Please position yourselves on the training mat so that each one of you has enough space to move your arms around freely and not hit anybody else." People start milling around on the mat, trying to position themselves properly. Minutes pass, and finally, the trainer is so annoyed that he asks you to write a program that will position all the people correctly, hoping it will be quicker than letting them figure it out for themselves!

You are given the dimensions (width and length) of the mat on which the class takes place. For every student, there is a circular area she has to have for herself, with a radius equal to the reach of her arms. These circles can not intersect, though they can touch; and the center of each circle (where the student stands) has to be on the mat. Note that the arms can reach outside the mat. You know that there's plenty of space on the mat — the area of the mat is at least five times larger than the total area of the circles required by all the people in the class. It will always be possible for all the people to position themselves as required.

Input

The first line contains three integers: N, W, and L ($1 \le N \le 10^3$, $1 \le W, L \le 10^9$), denoting the number of students, the width of the mat, and the length of the mat, respectively. The second line contains N integers r_i ($1 \le r_i \le 10^5$), denoting the reach of the arms of the i-th student. The area of the mat is at least 5 times larger than the total area of the circles.

Output

Output $2N$ real numbers $x_1, y_1, \ldots, x_N, y_N$, where the pair (x_i, y_i) is the position where the i-th student should stand with $0 \le x_i \le W$ and $0 \le y_i \le L$.

input
```
2 6 6
1 1
```
output
```
0.0 0.0
6.0 6.0
```

input
```
3 320 2
4 3 2
```
output
```
0.0 0.0
7.0 0.0
12.0 0.0
```

G. Persistent priority queue

2 seconds, 256 megabytes

You have to implement a data structure for storing multisets of integers. Initially, you have one empty multiset S_0. Then you have to perform n queries, query with index i ($1 \le i \le n$) results in creation of a new multiset S_i depending on the query type:

1. `1 v x` indicates that $S_i = S_v \cup \{x\}$. After performing this query you have to output the minimum element in S_i;
2. `2 u v` indicates that $S_i = S_u \cup S_v$. After performing this query you have to output the minimum element in S_i;
3. `3 v` indicates that you have to create $S_i = S_v$. Then, if S_i is empty, you have to output the word `empty`. Otherwise, you have to output the minimum element in S_i and then immediately remove it from S_i.

Input

The first line contains one integer n — the number of queries ($1 \le n \le 200\,000$).

To make sure you actually perform all queries online, you will have to maintain a special variable s, that is initially set to 0. After each query you have to update the value of s to $s_{new} = (s + x) \bmod 239017$, where x indicates the output of the query (or 0 for `empty`).

The next n lines describe queries, where i-th query has one of the 3 formats below:

1. `1 a b` indicates a query of the first type, where $v = (a + s) \bmod i$ and $x = (b + 17s) \bmod (10^9 + 1)$;
2. `2 a b`, indicates a query of the second type, where $u = (a + s) \bmod i$ and $v = (b + 13s) \bmod i$;
3. `3 a`, indicates a query of the third type, where $v = (a + s) \bmod i$.

It is guaranteed that the size of any multiset will not exceed 2^{63}.

Output

For each query output one line, containing the output of the query: either an integer or the word `empty`.

input
```
9
1 0 2
1 0 999999970
2 2 0
3 0
2 4 4
3 0
3 0
3 0
3 8
```
output
```
2
3
2
2
2
2
2
3
empty
```

H. Public Transit

4 seconds, 512 megabytes

The city of Treeonto is a tree of n vertices numbered from 0 to $n - 1$. Traveling along an edge in the graph takes one minute.

The citizens of Treeonto are upset that it takes too long to get around, so they have decided to build a teleporter. The teleporter will consist of two identical booths, each located in some vertex. If a citizen enters either booth, he or she may choose to teleport to the other booth instantly. It is allowed to build both booths in the same vertex.

We define the distance between two vertices as the smallest number of minutes needed to get from one vertex to the other. Let d be the maximum distance between any two vertices. Find the number of ways to place the teleporter such that d does not exceed x.

Input

The first line contains an integer n $(1 \le n \le 2000)$ — the number of vertices.

The second line contains $n-1$ integers, where each integer is between 0 and i, inclusive, representing the vertex to which vertex $i+1$ is connected.

The third line contains an integer x $(0 \le x \le n)$.

Output

Print a single integer, denoting the number of ways to place the teleporter such that the maximum distance between any two vertices does not exceed x.

input
```
4
0 1 2
1
```
output
```
1
```

input
```
3
0 0
2
```
output
```
6
```

input
```
6
0 0 0 1 1
2
```
output
```
1
```

input
```
7
0 1 0 1 2 4
3
```
output
```
0
```

input
```
16
0 1 0 2 0 0 4 5 8 9 10 11 8 4 6
7
```
output
```
31
```

In the second example, any teleporter placement is valid.

I. Desert

2 seconds, 256 megabytes

The route from Bythad do Bytara leads among the sands of the Great Byteotian Desert. Making the trip is trying, especially since there are only s wells along the route. Knowing that his country's prosperity depends on communications routes, the ruler of Byteotia has decided to have new wells dug along this particular route. The distance from Bythad to Bytara is $n+1$ bytemiles, and at each integer multiple of a bytemile from Bythad, there already is a well or a new one can be dug. However, the lower the water level is, the harder and costlier it is to dig a well.

Thus, the rules has commissioned the royal geologist Byteasar with surveying the options. Byteasar has m measurements obtained through the satellite network. Unfortunately, the information gathered by the satellites does not provide direct information about the water levels. Each measurement is for a contiguous fragment of the route and tells only that in certain points of this fragment, the water level is lower than in the remaining points. Moreover, it is known that the water level in every point is between 1 and 10^9 bytemeters below surface.

Help Byteasar by determining what the water level might be in every point along the route. It could turn out that the satellite data are contradictory.

Input

The first line of the standard input contains three integers, n, s, and m ($1 \le s \le n \le 100\,000$, $1 \le m \le 200\,000$), separated by single spaces, which specify the number of potential well slots, number of wells along the route and the number of satellite measurements.

The s lines that follow describe the wells: the i-th one contains two integers, p_i and d_i $(1 \le p_i \le n, 1 \le d_i \le 1\,000\,000\,000)$, which indicate that the i-th well is located p_i bytemiles from Bythad and is d_i bytometers deep (i.e., that the water level in the well is d_i bytometers below ground surface). The wells are given in an increasing order of p_i.

The next m lines describe the satellite measurements: the i-th one contains three integers l_i, r_i, and k_i $(1 \le l_i < r_i \le n$, $1 \le k_i \le r_i - l_i)$, followed by a sequence of k_i integers, $x_1, x_2, \ldots, x_{k_i}$ $(l_i \le x_1 < x_2 < \ldots < x_{k_i} \le r_i)$. These specify that a measurement was taken on the segment from l_i to r_i (including these endpoints), and it indicated that the water level in each of the points $x_1, \ldots, x_{k_i}$ is strictly lower than the water level in each of the remaining (integer) points of the interval, i.e. $\{l_i, \ldots, r_i\} \setminus \{x_1, \ldots, x_{k_i}\}$. The sum of all the k_i's does not exceed $200\,000$.

Output

If the measurements are contradictory, the first line of the standard output should contain a single word "NIE". Otherwise, the first line of the output should contain the word "TAK, whereas the second line should contain a sequence of n integers, each in the range from 1 to $1\,000\,000\,000$, specifying the depths below the surface of the water levels at successive points along the route from Bythad. If there are multiple solutions, your program should pick one arbitrarily.

input
```
5 2 2
2 7
5 3
1 4 2 2 3
4 5 1 4
```
output
```
TAK
6 7 1000000000 6 3
```

J. Array Transformer

3 seconds, 512 megabytes

You have an array of integers $a_1, \ldots a_n$. You need to perform q types of operations on the array.

There are four types of operations:

1. $1\ l\ r\ c$ — add c to each element a_i where $l \le i \le r$. Note that c can be negative.
2. $2\ l\ r\ d$ — replace each element a_i $(l \le i \le r)$ with $\left\lfloor \frac{a_i}{d} \right\rfloor$. Here, $\lfloor x \rfloor$ is the greatest integer y such that $y \le x$ (e.g., $\lfloor -2.5 \rfloor = -3$ and

$\lfloor -7 \rfloor = -7$).

3. $3\ l\ r$ — output the minimum value of a_i for $l \le i \le r$.
4. $4\ l\ r$ — output the sum of a_i for $l \le i \le r$.

Input

The first line contains two integer n and q — the length of the array and the number of operations.

The second line contains n integers a_i — the initial values of the array.

Each of the next q lines describes an operation in the format described above.

Constraints

$1 \le n, q \le 100000$

$-10^9 \le a_i \le 10^9$

$1 \le l \le r \le n$

$-10000 \le c \le 10000$

$2 \le d \le 10^9$

Output

For operations of types 3 and 4 output the answer, each on a new line.

input
```
10 9
-5 -4 -3 -2 -1 0 1 2 3 4
1 1 10 1
2 1 10 3
3 1 10
4 1 10
3 1 2
4 3 4
3 5 6
4 7 8
3 9 10
```
output
```
-2
-2
-2
-2
0
1
1
```

After the first operation the array will be [-4, -3, -2, -1, 0, 1, 2, 3, 4, 5].

After the second operation the array will be [-2, -1, -1, -1, 0, 0, 0, 1, 1, 1].

K. Semi Perfect Powers

2 seconds, 512 megabytes

Magical Girl Iris loves perfect powers. A positive integer n is a perfect power if and only if there are positive integers $b > 1$ and $c > 1$ such that $b^c = n$. For example, 8 $(= 2^3)$ and 243 $(= 3^5)$ are perfect powers, while 1 and 54 are not.

One day, Iris discovered that there are very few perfect powers. To avoid being disappointed, she quickly invented the semi-perfect powers: A positive integer n is a semi-perfect power if and only if there are positive integers $a \ge 1$, $b > 1$, and $c > 1$ such that $a < b$ and $a \cdot (b^c) = n$. For example, 243 $(= 1 \cdot 3^5)$ and 54 $(= 2 \cdot 3^3)$ are semi-perfect powers, while 1 and 24 are not.

Note that for some semi-perfect numbers there may be more than one corresponding triple (a, b, c). For example, 432 can be expressed as $2 \cdot 6^3$, but also as $3 \cdot 12^2$.

You are given longs L and R. Calculate and return the number of semi-perfect powers that lie between L and R, inclusive.

Input

The first line contains two integers L and R $(1 \le L \le R \le 8 \times 10^{16})$.

Output

Print a single integer, denoting the number of semi-perfect powers that lie between L and R, inclusive.

input
```
18 58
```
output
```
9
```

input
```
1 10
```
output
```
3
```

input
```
60 70
```
output
```
1
```

input
```
319268319114310 35860463407469139
```
output
```
95023825161
```

input
```
1 80000000000000000
```
output
```
169502909978
```

- **Example 1 Explanation:** There are 9 semi-perfect powers between 18 and 58, inclusive: 18, 25, 27, 32, 36, 48, 49, 50, and 54.
- **Example 2 Explanation:** Note that 1 is not considered to be a semi-perfect power.
- **Example 3 Explanation:** The number 64 is the only semi-perfect power in the given range. Note that there are multiple ways to choose a, b, and c when showing that 64 is a semi-perfect power. Still, each semi-perfect power should only be counted once.

L. Sortish1

6 seconds, 256 megabytes

Everyone likes some sequences more than others. Every person has their own function which tells them how good a sequence is. For example, for some people this function could simply be the count of negative numbers in the sequence. Jezalb's most favorite sequences are ones that are sorted in increasing order. When he sees a sequence S, he immediately calculates the number of pairs of indexes $i < j$ such that $S[i] < S[j]$. He calls this number the "sortedness" of S. This morning Jezalb entered a classroom and saw a permutation of 1 through n on the blackboard. He quickly calculated its sortedness. He then left the classroom and forgot the permutation. He only remembered the sortedness he computed. Later that day Jezalb reentered the classroom and saw a sequence on the blackboard. The sequence was a permutation of 1 through n, but with some elements erased. You are given this sequence with n elements. Some of the elements in sequence may be 0, which indicates an erased number. Jezalb thinks that the sequence may have been obtained by erasing some elements of the sequence he saw during his first visit to the classroom. He would like to restore the erased elements. You are given the sortedness and the sequence. Calculate the number of ways in which he can fill in the missing elements into sequence in such a way that the sortedness of the obtained permutation will be exactly sortedness.

Input

First line contains two integers n and m $(1 \le n \le 2000$, $0 \le m \le 10^9)$ — number of elements in the permutation and sortedness of the permutation, respectively.

Second line contains n integers a_i $(0 \le a_i \le n)$ — given sequence.

Positive elements in sequence will be distinct.

Number of elements equal to 0 in sequence will be between 0 and 14, inclusive.

Output

Output single number — answer to the problem.

input
```
5 5
4 0 0 2 0
```
output
```
2
```

input
```
4 4
0 0 0 0
```
output
```
5
```

input
```
3 2
1 3 2
```
output
```
1
```

input
```
6 3
0 0 2 0 0 0
```
output
```
4
```

input
```
2 87
2 0
```
output
```
0
```

input
```
10 30
0 6 3 0 0 2 10 0 0 0
```
output
```
34
```

input
```
23 100
0 13 0 0 12 0 0 0 2 0 0 10 5 0 0 0 0 0 0 7 15 16 20
```
output
```
53447326
```

In the first sample test there are six ways to fill in the missing elements. Out of those six permutations, only two have sortedness 5: {4, 1, 5, 2, 3} and {4, 3, 1, 2, 5}.

In the second sample test all 5 possible ways are: {1, 3, 4, 2}, {1, 4, 2, 3}, {2, 1, 4, 3}, {2, 3, 1, 4}, {3, 1, 2, 4}.

M. Yet Another Board Game

2 seconds, 512 megabytes

Manao is playing a new board game. The game is played on an $N \times M$ board with each cell initially colored either black or white. The cell on the intersection of the i-th (0-based) row and j-th (0-based) column is referred to as (i, j).

Manao may perform two types of moves:

- Pick a cell (i, j) $(0 \le i < N, 0 \le j < M)$ and toggle the color of cells $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$. If some of these cells are outside the board, the move is considered valid, and the cells outside of the board are ignored.
- Pick a cell (i, j) $(0 \le i < N, 0 \le j < M)$ and toggle the color of cells (i, j), $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$. Again, the cells outside of the board, if any, are ignored.

We call the two move types "type 1 moves" and "type 2 moves". In both cases, we say that Manao performed the move on the cell (i, j).

Manao cannot perform the moves arbitrarily, he has to follow some additional constraints: For each row, all moves applied to cells in the row have to be of the same type. Also, for each column, all moves applied to cells in the column have to be of the same type. In particular, Manao is not allowed to perform a type 1 move on a cell and then a type 2 move on the same cell (nor vice versa).

You are given a board consisting of N elements, each M characters long. The j-th character in the i-th row (0-based indices) of the board is 'W' if cell (i, j) is initially white, and 'B' otherwise. Manao wants to turn the board all black. Determine the minimum number of moves he needs to perform to accomplish this task. If it is impossible to turn every cell on the board black, return -1.

Input

The first line of input contains two integers N and M $(1 \le N, M \le 13)$, the number of rows and columns of the board, respectively.

The next N lines each contain a string of length M consisting of the characters 'W' and 'B'.

Output

Output a single integer, the minimum number of moves required to turn all cells on the board black. If it is impossible, output -1.

input
```
6 9
BBBBBBBBB
BBWBBBBBB
BWWWBBBBB
BBWBBBWBB
BBBBBWBWB
BBBBBBWBB
```
output
```
2
```

input
```
3 3
BBW
WWW
BWW
```
output
```
2
```

input
```
3 3
WBW
BBW
WBW
```
output
```
4
```

input
```
4 4
BBBB
WBWB
BBBB
BBBB
```

output

```
-1
```

input

```
3 7
WWWWWBW
WBWBWBW
BBWBBWW
```

output

```
7
```

input

```
10 10
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
```

output

```
30
```