CODEFORCES β
Sponsored by Telegram

Codeforces Round #405 (rated, Div. 1, based on VK Cup 2017 Round 1)

A. Bear and Different Names

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

In the army, it isn't easy to form a group of soldiers that will be effective on the battlefield. The communication is crucial and thus no two soldiers should share a name (what would happen if they got an order that Bob is a scouter, if there are two Bobs?).

A group of soldiers is effective if and only if their names are different. For example, a group (John, Bob, Limak) would be effective, while groups (Gary, Bob, Gary) and (Alice, Alice) wouldn't.

You are a spy in the enemy's camp. You noticed n soldiers standing in a row, numbered 1 through n. The general wants to choose a group of k consecutive soldiers. For every k consecutive soldiers, the general wrote down whether they would be an effective group or not.

You managed to steal the general's notes, with $n - k + 1$ strings $s_1, s_2, ..., s_{n-k+1}$, each either "YES" or "NO".

- The string s_1 describes a group of soldiers 1 through k ("YES" if the group is effective, and "NO" otherwise).
- The string s_2 describes a group of soldiers 2 through $k + 1$.
- And so on, till the string s_{n-k+1} that describes a group of soldiers $n - k + 1$ through n.

Your task is to find possible names of n soldiers. Names should match the stolen notes. Each name should be a string that consists of between 1 and 10 English letters, inclusive. The first letter should be uppercase, and all other letters should be lowercase. Names don't have to be existing names — it's allowed to print "Xyzzzdj" or "T" for example.

Find and print any solution. It can be proved that there always exists at least one solution.

Input

The first line of the input contains two integers n and k $(2 \le k \le n \le 50)$ — the number of soldiers and the size of a group respectively.

The second line contains $n - k + 1$ strings $s_1, s_2, ..., s_{n-k+1}$. The string s_i is "YES" if the group of soldiers i through $i + k - 1$ is effective, and "NO" otherwise.

Output

Find any solution satisfying all given conditions. In one line print n space-separated strings, denoting possible names of soldiers in the order. The first letter of each name should be uppercase, while the other letters should be lowercase. Each name should contain English letters only and has length from 1 to 10.

If there are multiple valid solutions, print any of them.

Examples

input
```
8 3
NO NO YES YES YES NO
```
output
```
Adam Bob Bob Cpqepqwer Limak Adam Bob Adam
```

input
```
9 8
YES NO
```
output
```
R Q Ccccccccc Ccocc Ccc So Strong Samples Ccc
```

input
```
3 2
NO NO
```
output
```
Na Na Na
```

Note

In the first sample, there are 8 soldiers. For every 3 consecutive ones we know whether they would be an effective group. Let's analyze the provided sample output:

- First three soldiers (i.e. Adam, Bob, Bob) wouldn't be an effective group because there are two Bobs. Indeed, the string s_1 is "`NO`".
- Soldiers 2 through 4 (Bob, Bob, Cpqepqwer) wouldn't be effective either, and the string s_2 is "`NO`".
- Soldiers 3 through 5 (Bob, Cpqepqwer, Limak) would be effective, and the string s_3 is "`YES`".
- ...,
- Soldiers 6 through 8 (Adam, Bob, Adam) wouldn't be effective, and the string s_6 is "`NO`".

B. Bear and Tree Jumps

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

A tree is an undirected connected graph without cycles. The distance between two vertices is the number of edges in a simple path between them.

Limak is a little polar bear. He lives in a tree that consists of n vertices, numbered 1 through n.

Limak recently learned how to jump. He can jump from a vertex to any vertex within distance at most k.

For a pair of vertices (s, t) we define $f(s, t)$ as the minimum number of jumps Limak needs to get from s to t. Your task is to find the sum of $f(s, t)$ over all pairs of vertices (s, t) such that $s < t$.

Input

The first line of the input contains two integers n and k ($2 \le n \le 200\,000$, $1 \le k \le 5$) — the number of vertices in the tree and the maximum allowed jump distance respectively.

The next $n - 1$ lines describe edges in the tree. The i-th of those lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$) — the indices on vertices connected with i-th edge.

It's guaranteed that the given edges form a tree.

Output

Print one integer, denoting the sum of $f(s, t)$ over all pairs of vertices (s, t) such that $s < t$.

Examples

input

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

output

```
20
```

input

```
13 3
1 2
3 2
4 2
5 2
3 6
10 6
6 7
6 13
5 8
5 9
9 11
11 12
```

output

```
114
```

input

```
3 5
2 1
3 1
```

output

```
3
```

Note

In the first sample, the given tree has 6 vertices and it's displayed on the drawing below. Limak can jump to any vertex within distance at most 2. For example, from the vertex 5 he can jump to any of vertices: 1, 2 and 4 (well, he can also jump to the vertex 5 itself).

There are pairs of vertices (s, t) such that $s < t$. For 5 of those pairs Limak would need two jumps: $(1, 6), (3, 4), (3, 5), (3, 6), (5, 6)$. For other 10 pairs one jump is enough. So, the answer is $5 \cdot 2 + 10 \cdot 1 = 20$.

In the third sample, Limak can jump between every two vertices directly. There are 3 pairs of vertices $(s < t)$, so the answer is $3 \cdot 1 = 3$.

C. Bear and Company

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

Bear Limak prepares problems for a programming competition. Of course, it would be unprofessional to mention the sponsor name in the statement. Limak takes it seriously and he is going to change some words. To make it still possible to read, he will try to modify each word as little as possible.

Limak has a string s that consists of uppercase English letters. In one move he can swap two **adjacent** letters of the string. For example, he can transform a string "ABBC" into "BABC" or "ABCB" in one move.

Limak wants to obtain a string without a substring "VK" (i.e. there should be no letter 'V' immediately followed by letter 'K'). It can be easily proved that it's possible for any initial string s.

What is the minimum possible number of moves Limak can do?

Input

The first line of the input contains an integer n $(1 \le n \le 75)$ — the length of the string.

The second line contains a string s, consisting of uppercase English letters. The length of the string is equal to n.

Output

Print one integer, denoting the minimum possible number of moves Limak can do, in order to obtain a string without a substring "VK".

Examples

input

```
4
VKVK
```

output

```
3
```

input

```
5
BVVKV
```

output

```
2
```

input

```
7
VVKEVKK
```

output

```
3
```

input

```
20
VKVKVVVKVOVKVQKKKVVK
```

output

```
8
```

input

```
5
LIMAK
```

output

```
0
```

Note

In the first sample, the initial string is "VKVK". The minimum possible number of moves is 3. One optimal sequence of moves is:

1. Swap two last letters. The string becomes "VKKV".
2. Swap first two letters. The string becomes "KVKV".
3. Swap the second and the third letter. The string becomes "KKVV". Indeed, this string doesn't have a substring "VK".

In the second sample, there are two optimal sequences of moves. One is "BVVKV" $\to$ "VBVKV" $\to$ "VVBKV". The other is "BVVKV" $\to$ "BVKVV" $\to$ "BKVVV".

In the fifth sample, no swaps are necessary.

D. Bear and Rectangle Strips

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Limak has a grid that consists of 2 rows and n columns. The j-th cell in the i-th row contains an integer $t_{i,j}$ which can be positive, negative or zero.

A non-empty rectangle of cells is called *nice* if and only if the sum of numbers in its cells is equal to 0.

Limak wants to choose some nice rectangles and give them to his friends, as gifts. No two chosen rectangles should share a cell. What is the maximum possible number of nice rectangles Limak can choose?

Input

The first line of the input contains an integer n $(1 \le n \le 300\,000)$ — the number of columns in the grid.

The next two lines contain numbers in the grid. The i-th of those two lines contains n integers $t_{i,1}, t_{i,2}, ..., t_{i,n}$ $(-10^9 \le t_{i,j} \le 10^9)$.

Output

Print one integer, denoting the maximum possible number of cell-disjoint nice rectangles.

Examples

input

```
6
70 70 70 70 70 -15
90 -60 -30 30 -30 15
```

output

```
3
```

input

```
4
0 -1 0 0
0 0 1 0
```

output

```
6
```

input

```
3
1000000000 999999999 -1000000000
999999999 -1000000000 -999999998
```

output

```
1
```

Note

In the first sample, there are four nice rectangles:

Limak can't choose all of them because they are not disjoint. He should take three nice rectangles: those denoted as blue frames on the drawings.

In the second sample, it's optimal to choose six nice rectangles, each consisting of one cell with a number 0.

In the third sample, the only nice rectangle is the whole grid — the sum of all numbers is 0. Clearly, Limak can choose at most one nice rectangle, so the answer is 1.

E. Bear and Isomorphic Points

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

Bearland is a big square on the plane. It contains all points with coordinates not exceeding 10^6 by the absolute value.

There are n houses in Bearland. The i-th of them is located at the point (x_i, y_i). The n points are distinct, but some subsets of them may be collinear.

Bear Limak lives in the first house. He wants to destroy his house and build a new one somewhere in Bearland.

Bears don't like big changes. For every three points (houses) p_i, p_j and p_k, the sign of their cross product $(p_j - p_i) \times (p_k - p_i)$ should be the same before and after the relocation. If it was negative/positive/zero, it should still be negative/positive/zero respectively. This condition should be satisfied for all triples of indices (i, j, k), possibly equal to each other or different than 1. Additionally, Limak isn't allowed to build the house at the point where some other house already exists (but it can be the point where his old house was).

In the formula above, we define the difference and the cross product of points (a_x, a_y) and (b_x, b_y) as:

$$(a_x, a_y) - (b_x, b_y) = (a_x - b_x, a_y - b_y),$$
$$(a_x, a_y) \times (b_x, b_y) = a_x \cdot b_y - a_y \cdot b_x.$$

Consider a set of possible new placements of Limak's house. Your task is to find the area of that set of points.

Formally, let's say that Limak chooses the new placement randomly (each coordinate is chosen independently uniformly at random from the interval $[-10^6, 10^6]$). Let p denote the probability of getting the allowed placement of new house. Let S denote the area of Bearland ($S = 4 \cdot 10^{12}$). Your task is to find $p \cdot S$.

Input

The first line of the input contains an integer T $(1 \le T \le 500)$ — the number of test cases. The description of the test cases follows.

The first line of the description of a test case contains an integer n $(3 \le n \le 200\,000)$ — the number of houses.

The i-th of the next n lines contains two integers x_i and y_i $(-10^6 \le x_i, y_i \le 10^6)$ — coordinates of the i-th house. No two houses are located at the same point in the same test case. Limak lives in the first house.

The sum of n won't exceed $200\,000$.

Output

Print one real value, denoting the area of the set of points that are possible new placements of Limak's house.

Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}. More precisely, let the jury's answer be b, and your answer be a. Then your answer will be accepted if and only if .

Example

input

```
4
4
5 3
0 1
10 1
3 51
3
-999123 700000
-950000 123456
-950000 987654
3
2 3
10 -1
-4 6
5
1 3
5 2
6 1
4 4
-3 3
```

output

```
250.000000000000
100000000000.000000000000
0.000000000000
6.562500000000
```

Note

In the sample test, there are 4 test cases.

In the first test case, there are four houses and Limak's one is in $(5, 3)$. The set of valid new placements form a triangle with vertices in points $(0, 1)$, $(10, 1)$ and $(3, 51)$, without its sides. The area of such a triangle is 250.

In the second test case, the set of valid new placements form a rectangle of width $50\,000$ and height $2\,000\,000$. Don't forget that the new placement must be inside the big square that represents Bearland.

In the third test case, the three given points are collinear. Each cross product is equal to 0 and it should be 0 after the relocation as well. Hence, Limak's new house must lie on the line that goes through the given points. Since it must also be inside the big square, new possible placements are limited to some segment (excluding the two points where the other houses are). The area of any segment is 0.