Problem A. Exam

Source file name:	exam.c, exam.cpp, exam.java, exam.py
Input:	`Standard`
Output:	`Standard`

Your friend and you took a true/false exam of n questions. You know your answers, your friend's answers, and that your friend got k questions correct.

Compute the maximum number of questions you could have gotten correctly.

Input

The first line of input contains a single integer k.

The second line contains a string of n ($1 \leq n \leq 1000$) characters, the answers you wrote down. Each letter is either a '**T**' or an '**F**'.

The third line contains a string of n characters, the answers your friend wrote down. Each letter is either a '**T**' or an '**F**'.

The input will satisfy $0 \leq k \leq n$.

Output

Print, on one line, the maximum number of questions you could have gotten correctly.

Example

Input	Output
3 FTFFF TFTTT	2
6 TTFTFFTFTF TTTTFFTTTT	9

Problem B. Coprime Integers

Source file name:	coprime.c, coprime.cpp, coprime.java, coprime.py
Input:	`Standard`
Output:	`Standard`

✓ $35 = 7 \times 5 \times 1$

✓ $39 = 3 \times 13 \times 1$

Given intervals $[a, b]$ and $[c, d]$, count the number of pairs of coprime integers (x, y) such that $a \leq x \leq b$ and $c \leq y \leq d$.

Two numbers are coprime if their greatest common divisor is 1. A divisor of a number is a positive integer that evenly divides that number.

Input

The input consists of a single line containing space-separated integers a, b, c, and d ($1 \leq a \leq b \leq 10^7$, $1 \leq c \leq d \leq 10^7$).

Output

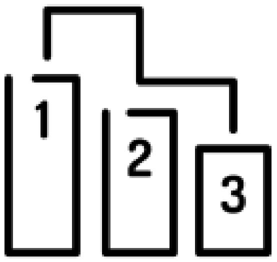
Print the result as a single integer on one line.

Example

Input	Output
`1 5 1 5`	`19`
`12 12 1 12`	`4`
`1 100 1 100`	`6087`

Problem C. Contest Setting

Source file name:	contest.c, contest.cpp, contest.java, contest.py
Input:	`Standard`
Output:	`Standard`



A group of contest writers have written n problems and want to use k of them in an upcoming contest. Each problem has a *difficulty level*. A contest is valid if all of its k problems have different *difficulty levels*.

Compute how many distinct valid contests the contest writers can produce. Two contests are distinct if and only if there exists some problem present in one contest but not present in the other.

Print the result modulo 998,244,353.

Input

The first line of input contains two space-separated integers n and k ($1 \le k \le n \le 1000$).

The next line contains n space-separated integers representing the difficulty levels. The difficulty levels are between 1 and 10^9 (inclusive).

Output

Print the number of distinct contests possible, modulo 998,244,353.

Example

Input	Output

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

`5 2`
`1 1 1 2 2`
	`6`

`12 5`
`3 1 4 1 5 9 2 6 5 3 5 8`
	`316`

Problem D. Count The Bits

Source file name:	bits.c, bits.cpp, bits.java, bits.py
Input:	`Standard`
Output:	`Standard`

Given an integer k and a number of bits b ($1 \leq b \leq 128$), calculate the total number of 1 bits in the binary representations of multiples of k between 0 and $2^b - 1$ (inclusive), modulo $10^9 + 9$ ($1,000,000,009$).

Input

The input will consist of two integers k and b on a single line, with $1 \leq k \leq 1000$ and $1 \leq b \leq 128$.

Output

Write your result as an integer on a single line.

Example

Input	Output
`1 4`	`32`
`10 5`	`8`
`100 7`	`3`
`3 28`	`252698795`

Problem E. Cops And Robbers

Source file name:	cops.c, cops.cpp, cops.java, cops.py
Input:	`Standard`
Output:	`Standard`

The First Universal Bank of Denview has just been robbed! You want to catch the robbers before they leave the state of Calirado.

The state of Calirado is a perfect grid of size m-by-n. The robbers will try to escape by going from grid square to grid square (only through edges, not corners) until they reach the border of the state.

You can barricade some grid squares to stop the robbers from using them. However, depending on the terrain, different grid squares may cost different amount of money, and some grid squares may not be barricaded at all!

Find the cheapest set of grid squares to barricade that guarantees no escape for the robbers.

Input

The first line of input contains three space-separated integers n, m, and c ($1 \leq n, m \leq 30$, and $1 \leq c \leq 26$), where n and m are the dimensions of the grid, and c is the number of types of terrain in Calirado.

Each of the next m lines contains n letters each, representing the grid.

- Character '**B**' indicates the First Universal Bank of Denview; it is where the robbers currently are.
- Characters '**a**' through '**z**' represent the different types of terrain. Only the first c alphabets will appear in the grid.
- A dot ('**.**') represents a grid square that cannot be barricaded.

It is guaranteed that the grid will contain exactly one **B** character.

Finally, the last line of the input contains c space-separated integers between 1 and 100,000 (inclusive), representing the cost of barricading a single grid square of type '**a**', '**b**', and so on.

Output

Print, on one line, the minimum total cost of barricading plan that guarantees no exit for the robbers.

If there is no way to prevent the robbers from escaping, print **-1** instead.

In the first example, the minimum cost is to barricade the central three squares on each side for a total cost of 12.

In the second example, since the bank is on the border, and we cannot barricade the bank, there is no way to prevent the robbers from escaping the state.

Example

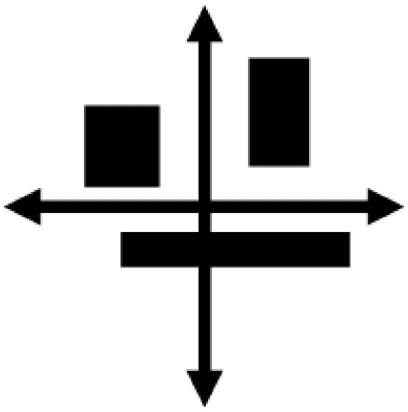
Input	Output

`5 5 1`
`aaaaa`
`a...a`
`a.B.a`
`a...a`
`aaaaa`
`1`
	`12`

`2 2 1`
`aB`
`aa`
`1`
	`-1`

Problem F. Rectangles

Source file name:	rectangles.c, rectangles.cpp, rectangles.java, rectangles.py
Input:	`Standard`
Output:	`Standard`



You are given several axis-aligned rectangles. Compute the sum of the area of the regions that are covered by an odd number of rectangles.

Input

The first line of input contains a single integer n ($1 \leq n \leq 10^5$), representing the number of rectangles.

Each of the next n lines contains four space-separated integers x_1, y_1, x_2, and y_2, each between 0 and 10^9, describing the coordinates of a rectangle.

Output

Print, on one line, the total area covered by an odd number of rectangles as an exact integer.

Example

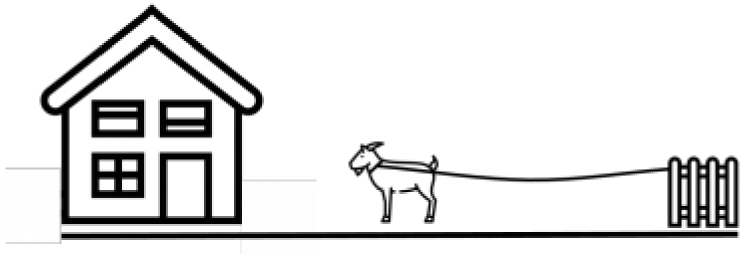
Input	Output

2
0 0 4 4
1 1 3 3
	12

4
0 0 10 10
1 1 11 11
2 2 12 12
3 3 13 13
	72

Problem G. Goat on a Rope

Source file name:	goaton.c, goaton.cpp, goaton.java, goaton.py
Input:	`Standard`
Output:	`Standard`



You have a house, which you model as an axis-aligned rectangle with corners at (x_1, y_1) and (x_2, y_2).

You also have a goat, which you want to tie to a fence post located at (x, y), with a rope of length l. The goat can reach anywhere within a distance l from the fence post.

Find the largest value of l so that the goat cannot reach your house.

Input

Input consists of a single line with six space-separated integers x, y, x_1, y_1, x_2, and y_2. All the values are guaranteed to be between -1000 and 1000 (inclusive).

It is guaranteed that $x_1 < y_1$ and $x_2 < y_2$, and that (x, y) lies strictly outside the axis-aligned rectangle with corners at (x_1, y_1) and (x_2, y_2).

Output

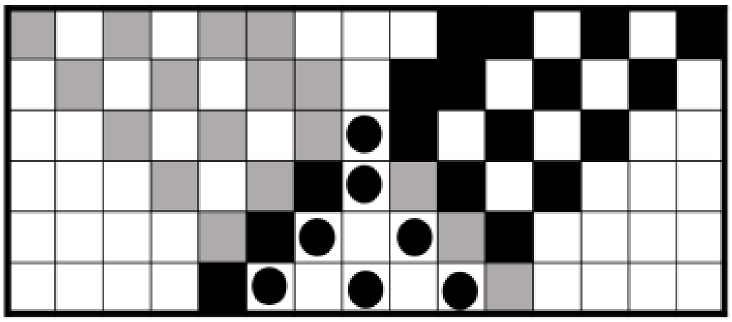
Print, on one line, the maximum value of l, rounded and displayed to exactly three decimal places.

Example

Input	Output
`7 4 0 0 5 4`	`2.000`
`6 0 0 2 7 6`	`2.000`
`4 8 7 8 9 9`	`3.000`

Problem H. Repeating Goldbachs

Source file name:	goldbachs.c, goldbachs.cpp, goldbachs.java, goldbachs.py
Input:	`Standard`
Output:	`Standard`



The Goldbach Conjecture states that any even number $x \geq 4$ can be expressed as the sum of two primes. It can be verified that the conjecture is true for all $x \leq 10^6$.

Define a Goldbach step as taking x ($4 \leq x \leq 10^6$), finding primes p and q (with $p \leq q$) that sum to x, and replacing x with $q - p$. If there are multiple pairs of primes which sum to x, we take the pair with the largest difference. That difference must be even and less than x. Therefore, we can repeat more Goldbach steps, until we can reach a number less than 4.

Given x, find how many Goldbach steps it takes until reaching a number less than 4.

Input

The input will consist of a single integer x ($4 \leq x \leq 10^6$).

Output

Print, on a single line, the number of Goldbach steps it takes to reach a number less than 4.

Example

Input	Output
20	3
30	4
40	5
50	6
60	7
70	8

Problem I. Inversions

Source file name:	inversions.c, inversions.cpp, inversions.java, inversions.py
Input:	`Standard`
Output:	`Standard`

You are given a sequence $p = (p_1, \ldots, p_n)$ of n integers between 1 and k, inclusive. Some of the integers are not determined, but they must still be between 1 and k.

An inversion is a pair of indices (i, j) with $i < j$ and $p_i > p_j$.

Find the maximum possible number of inversions in p obtained by replacing the undetermined numbers with values between 1 and k. Note that the numbers can be duplicated.

Input

The first line of input contains two space-separated integers n and k ($1 \le n \le 2 \cdot 10^5$ and $1 \le k \le 100$). The next n lines describe the sequence p, and each contains a single integer between 0 and k, inclusive. The value 0 represents an undetermined number.

Output

Print, on a single line, the maximum possible number of inversions.

Example

Input	Output

6 9
0
8
4
3
0
0
	15

10 9
5
2
9
0
7
4
8
7
0
0
	28

10 9
7
4
0
0
8
5
0
0
3
1
	36

Problem J. Time Limits

Source file name:	timelimits.c, timelimits.cpp, timelimits.java, timelimits.py
Input:	`Standard`
Output:	`Standard`

A contest setter wants to determine the time limits for a given problem. There are n model solutions, and solution k takes t_k milliseconds to run on the test data. The contest setter wants the time limit to be an integer number of seconds, and wants the time limit to be at least s times larger than the slowest model solution. Compute the minimum time limit the contest setter can set.

Input

The first line of input contains two space-separated integers n and s ($1 \leq n \leq 100$ and $1 \leq s \leq 20$).

The second line of input contains n space-separated integers $t_1, \ldots, t_n$ ($1 \leq t_k \leq 2000$ for all $k = 1, \ldots, n$).

Output

Print, on one line, the minimum time limit (in seconds) as a single integer.

Example

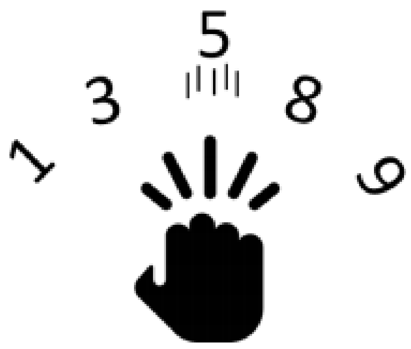
Input	Output

`2 5`
`200 250`
	`2`

`3 4`
`47 1032 1107`
	`5`

Problem K. Knockout

Source file name:	knockout.c, knockout.cpp, knockout.java, knockout.py
Input:	`Standard`
Output:	`Standard`



The solitaire game Knockout is played as follows. The digits from 1 to 9 are written down on a board, in order. In each turn, you throw a pair of six-sided dice. You sum the dice and cross out some set of digits that sum to the same total. If you cannot, the game ends and your score is the concatenation of the remaining digits. Otherwise, you throw the dice again and continue.

This game can be played to either minimize or maximize your score. Given a position of the game (what digits remain) and a roll of the dice, compute which digits you should remove and what your expected total score is for both minimizing and maximizing versions of the game.

Input

The input contains a single line with three space-separated integers. The first is the board state, containing some nonempty subset of the digits 1 through 9, in order. The next two integers are numbers between 1 to 6 (inclusive), representing the roll of the two dice.

Output

On the first line of output, print the digit(s) you should remove to minimize the expected score, followed by the expected score. On the second line of output, do the same for the maximizing version of Knockout.

If multiple digits are removed, list the digits in order with no space separating them. If you cannot remove digits to match the sum of the dice, print **-1** for your move instead.

The expected scores should be printed with exactly five digits after the decimal point.

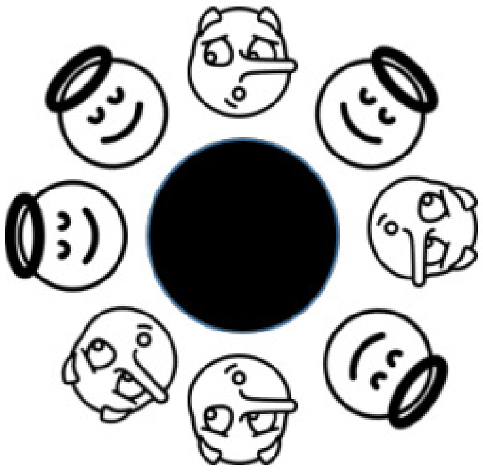
Example

Input	Output
`1345 1 1`	`-1 1345.00000` `-1 1345.00000`

`12349 3 1`	`13 151.70370`
	`4 401.24546`

Problem L. Liars

Source file name:	liars.c, liars.cpp, liars.java, liars.py
Input:	`Standard`
Output:	`Standard`



There are n people in a circle, numbered from 1 to n, each of whom always tells the truth or always lies.

Each person i makes a claim of the form: "the number of truth-tellers in this circle is between a_i and b_i, inclusive."

Compute the maximum number of people who could be telling the truth.

Input

The first line contains a single integer n ($1 \leq n \leq 10^3$). Each of the next n lines contains two space-separated integers a_i and b_i ($0 \leq a_i \leq b_i \leq n$).

Output

Print, on a single line, the maximum number of people who could be telling the truth. If the given set of statements is inconsistent, print **-1** instead.

Example

Input	Output

3
1 1
2 3
2 2
	2

8
0 1
1 7
4 8
3 7
1 2
4 5
3 7
1 8
	-1

Problem M. Mobilization

Source file name:	mobilization.c, mobilization.cpp, mobilization.java, mobilization.py
Input:	`Standard`
Output:	`Standard`

In some strategy games you are required to mobilize an army. There are n troops to choose from, each of which has a unit cost c_i, health h_i, and potency p_i. You can acquire any combination of the troop types (even fractional units), such that the total cost is no more than C. The *efficacy* of the army is equal to its total health value, multiplied by its total potency. What is the greatest efficacy you can achieve given the troops available?

You may assume that there are always sufficient troops to buy as many as you want (subject to the total cost constraint).

Input

The first of input consists of two space-separated integers n and C ($1 \leq n \leq 3 \cdot 10^4$ and $1 \leq C \leq 10^5$).

Each of the next n lines starts with an integer c_i ($1 \leq c_i \leq 10^5$), followed by two decimal values h_i and p_i ($0.0 \leq hi, pi \leq 1.0$). The numbers are space-separated.

Output

Print, on a single line, the greatest possible efficacy, with exactly two digits after the decimal point.

Example

Input	Output

`4 100000`
`300 1 0.02`
`500 0.2 1`
`250 0.3 0.1`
`1000 1 0.1`
	`19436.05`

`2 100`
`1 0.1 1`
`1 1 0.1`
	`3025.00`