CODEFORCES β
Sponsored by Telegram

Codeforces Round #456 (Div. 2)

A. Tricky Alchemy

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

During the winter holidays, the demand for Christmas balls is exceptionally high. Since it's already 2018, the advances in alchemy allow easy and efficient ball creation by utilizing magic crystals.

Grisha needs to obtain some yellow, green and blue balls. It's known that to produce a **yellow** ball one needs two yellow crystals, **green** — one yellow and one blue, and for a **blue** ball, three blue crystals are enough.

Right now there are A yellow and B blue crystals in Grisha's disposal. Find out how many additional crystals he should acquire in order to produce the required number of balls.

Input

The first line features two integers A and B $(0 \le A, B \le 10^9)$, denoting the number of yellow and blue crystals respectively at Grisha's disposal.

The next line contains three integers x, y and z $(0 \le x, y, z \le 10^9)$ — the respective amounts of yellow, green and blue balls to be obtained.

Output

Print a single integer — the minimum number of crystals that Grisha should acquire in addition.

Examples

input

```
4 3
2 1 1
```

output

```
2
```

input

```
3 9
1 1 3
```

output

```
1
```

input

```
12345678 87654321
43043751 1000000000 53798715
```

output

```
2147483648
```

Note

In the first sample case, Grisha needs five yellow and four blue crystals to create two yellow balls, one green ball, and one blue ball. To do that, Grisha needs to obtain two additional crystals: one yellow and one blue.

B. New Year's Eve

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Since Grisha behaved well last year, at New Year's Eve he was visited by Ded Moroz who brought an enormous bag of gifts with him! The bag contains n sweet candies from *the good ol' bakery*, each labeled from 1 to n corresponding to its tastiness. No two candies have the same tastiness.

The choice of candies has a direct effect on Grisha's happiness. One can assume that he should take the tastiest ones — but no, the holiday magic turns things upside down. It is the xor-sum of tastinesses that matters, not the ordinary sum!

A xor-sum of a sequence of integers $a_1, a_2, ..., a_m$ is defined as the bitwise XOR of all its elements: $a_1 \oplus a_2 \oplus \ldots \oplus a_m$, here $\oplus$ denotes the bitwise XOR operation; more about bitwise XOR can be found here.

Ded Moroz warned Grisha he has more houses to visit, so Grisha can take **no more than k** candies from the bag. Help Grisha determine the largest xor-sum (largest xor-sum means maximum happiness!) he can obtain.

Input

The sole string contains two integers n and k ($1 \leq k \leq n \leq 10^{18}$).

Output

Output one number — the largest possible xor-sum.

Examples

input Copy

```
4 3
```

output

```
7
```

input Copy

```
6 6
```

output

```
7
```

Note

In the first sample case, one optimal answer is 1, 2 and 4, giving the xor-sum of 7.

In the second sample case, one can, for example, take all six candies and obtain the xor-sum of 7.

C. Perun, Ult!

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

A lot of students spend their winter holidays productively. Vlad has advanced very well in doing so! For three days already, fueled by salads and tangerines — the leftovers from New Year celebration — he has been calibrating his rating in his favorite MOBA game, playing as a hero named Perun.

Perun has an ultimate ability called "Thunderwrath". At the instant of its activation, each enemy on the map (n of them in total) loses $damage$ health points as a single-time effect. It also has a restriction: it can only activated when the moment of time is an **integer**. The initial bounty for killing an enemy is $bounty$. Additionally, it increases by $increase$ each second. Formally, if at some second t the ability is activated and the i-th enemy is killed as a result (i.e. his health drops to zero or lower), Vlad earns $bounty + t \cdot increase$ units of gold.

Every enemy can receive damage, as well as be healed. There are multiple ways of doing so, but Vlad is not interested in details. For each of n enemies he knows:

- max_health_i — maximum number of health points for the i-th enemy;
- $start\_health_i$ — initial health of the enemy (on the 0-th second);
- $regen_i$ — the amount of health the i-th enemy can regenerate per second.

There also m health updates Vlad knows about:

- $time_j$ — time when the health was updated;
- $enemy_j$ — the enemy whose health was updated;
- $health_j$ — updated health points for $enemy_j$.

Obviously, Vlad wants to maximize his profit. If it's necessary, he could even wait for years to activate his ability at the right second. Help him determine the exact second (note that it must be **an integer**) from 0 (inclusively) to $+\infty$ so that a single activation of the ability would yield Vlad the maximum possible amount of gold, and print this amount.

Input

In the first line, two integers are given (separated by spaces) — n and m ($1 \le n \le 10^5$, $0 \le m \le 10^5$).

In the second line, there are three integers: $bounty$, $increase$ and $damage$ ($1 \le bounty, damage \le 10^9$, $0 \le increase \le 10^4$).

Each of the following n lines has three integers — max_health_i, $start\_health_i$, $regen_i$ ($1 \le start\_health_i \le max\_health_i \le 10^9$, $0 \le regen_i \le max\_health_i$).

The next m lines contain three integers each — $time_j$, $enemy_j$, $health_j$ ($1 \le time_j \le 10^9$, $1 \le enemy_j \le n$, $1 \le health_j \le max\_health_{enemy_j}$). It is guaranteed that there is no more than one hearth change per second for each enemy: more formally, for each a, b so that $1 \le a, b \le m$, $a \ne b$ holds that if $time_a = time_b$, then $enemy_a \ne enemy_b$.

Output

Output the single integer — the maximum amount of gold Vlad can obtain if he applies "Thunderwrath" exactly once, or -1 if this amount can be **infinitely** large.

Examples

input

```
3 2
1000 10 50
70 5 5
90 70 1
110 20 2
20 2 10
30 3 10
```

output

```
3000
```

input

```
1 1
500 50 1000
750 750 20
10 1 300
```

output

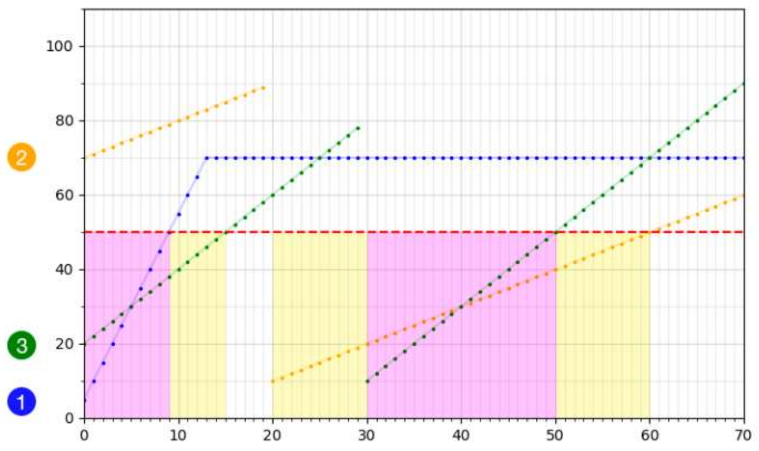
```
-1
```

Note

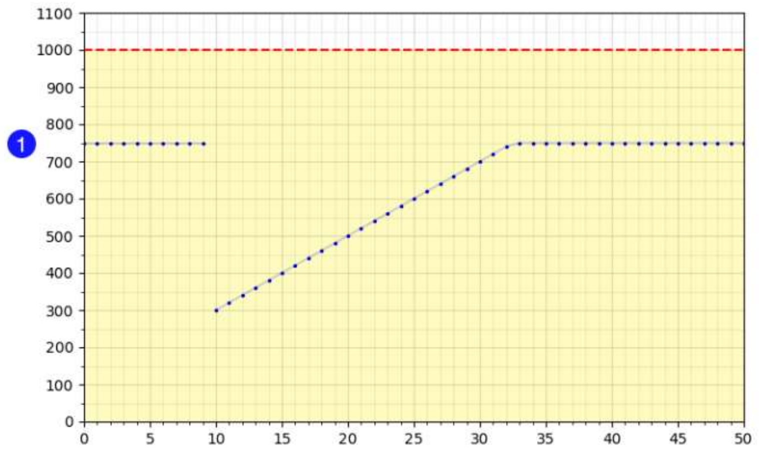
On the pictures you can see health points of each enemy versus time in sample cases.

Periods when Vlad can kill one enemy are marked with yellow color.

Periods when Vlad can kill two enemies are marked with purple color.



In the first sample case, Vlad can activate the ability at the 50-th second: the enemies 2 and 3 will die since they would have 40 and 50 health points correspondingly. Vlad will earn $2\cdot(1000 + 50\cdot 10) = 3000$ gold.



In the second sample case, the maximum amount of health for the enemy 1 is less than the damage dealt by the ability. Hence, the enemy could be killed anytime. As the bounty increases by 50 over the time, the maximum possible amount of gold is infinite.

D. Fishes

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

While Grisha was celebrating New Year with Ded Moroz, Misha gifted Sasha a small rectangular pond of size $n \times m$, divided into cells of size 1×1, inhabited by tiny evil fishes (no more than one fish per cell, otherwise they'll strife!).

The gift bundle also includes a square scoop of size $r \times r$, designed for fishing. If the lower-left corner of the scoop-net is located at cell (x, y), all fishes inside the square $(x, y)...(x + r - 1, y + r - 1)$ get caught. Note that the scoop-net should lie completely inside the pond when used.

Unfortunately, Sasha is not that skilled in fishing and hence throws the scoop randomly. In order to not frustrate Sasha, Misha decided to release k fishes into the empty pond in such a way that the expected value of the number of caught fishes is as high as possible. Help Misha! In other words, put k fishes in the pond into distinct cells in such a way that when the scoop-net is placed into a random position among $(n - r + 1) \cdot (m - r + 1)$ possible positions, the average number of caught fishes is as high as possible.

Input

The only line contains four integers n, m, r, k ($1 \le n, m \le 10^5$, $1 \le r \le min(n, m)$, $1 \le k \le min(n \cdot m, 10^5)$).

Output

Print a single number — the maximum possible expected number of caught fishes.

You answer is considered correct, is its absolute or relative error does not exceed 10^{-9}. Namely, let your answer be a, and the jury's answer be b. Your answer is considered correct, if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-9}$.

Examples

input

```
3 3 2 3
```

output

```
2.0000000000
```

input

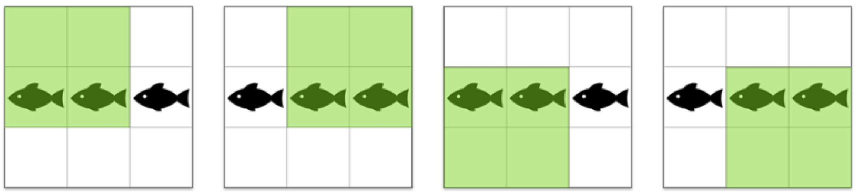
```
12 17 9 40
```

output

```
32.8333333333
```

Note

In the first example you can put the fishes in cells $(2, 1)$, $(2, 2)$, $(2, 3)$. In this case, for any of four possible positions of the scoop-net (highlighted with light green), the number of fishes inside is equal to two, and so is the expected value.



E. Prime Gift

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Opposite to Grisha's nice behavior, Oleg, though he has an entire year at his disposal, didn't manage to learn how to solve number theory problems in the past year. That's why instead of Ded Moroz he was visited by his teammate Andrew, who solemnly presented him with a set of n **distinct prime** numbers alongside with a simple task: Oleg is to find the k-th smallest integer, such that **all** its prime divisors are in this set.

Input

The first line contains a single integer n $(1 \le n \le 16)$.

The next line lists n distinct prime numbers $p_1, p_2, ..., p_n$ $(2 \le p_i \le 100)$ in ascending order.

The last line gives a single integer k $(1 \le k)$. It is guaranteed that the k-th smallest integer such that all its prime divisors are in this set does not exceed 10^{18}.

Output

Print a single line featuring the k-th smallest integer. It's guaranteed that the answer doesn't exceed 10^{18}.

Examples

input Copy

```
3
2 3 5
7
```

output

```
8
```

input Copy

```
5
3 7 11 13 31
17
```

output

```
93
```

Note

The list of numbers with all prime divisors inside $\{2, 3, 5\}$ begins as follows:

$$(1, 2, 3, 4, 5, 6, 8, ...)$$

The seventh number in this list $(1$-indexed) is eight.