Problem A. Leftmost Occurrences

Input file:	standard input
Output file:	standard output
Time limit:	5 seconds
Memory limit:	256 mebibytes

You are given sorted sequence of integer numbers $a_1, a_2, \ldots, a_n$ ($a_1 \le a_2 \le \cdots \le a_n$). Also you are given m numbers $b_1, b_2, \ldots, b_m$. For each b_j print such first (smallest) index i in the sequence a that $a_i = b_j$. Print `-1` if such index doesn't exist.

Prefer to submit your solution with PyPy if you use Python.

Input

The first line contains n ($1 \le n \le 2 \cdot 10^5$).

The second line contains integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$). The given integers are sorted by non-decreasing.

The third line contains m ($1 \le m \le 2 \cdot 10^5$).

The fourth line contains integers $b_1, b_2, \ldots, b_m$ ($-10^9 \le b_j \le 10^9$).

Output

Print the sequence of m integers, the j-th of them should be equal to such smallest i ($1 \le i \le n$) that $a_i = b_j$. If there is no such i, print `-1`.

Example

standard input
8 3 4 4 4 6 7 10 10 11 100 8 7 4 3 4 3 10 6 -100 2
standard output
-1 -1 6 2 1 2 1 7 5 -1 -1

$l = l \quad r = l + 1$

if $a[l]$

Problem B. Rightmost Occurrences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 megabytes

You are given sorted sequence of integer numbers $a_1, a_2, \ldots, a_n$ ($a_1 \le a_2 \le \cdots \le a_n$). Also you are given m numbers $b_1, b_2, \ldots, b_m$. For each b_j print such last (greatest) index i in the sequence a that $a_i = b_j$. Print `-1` if such index doesn't exist.

Prefer to submit your solution with PyPy if you use Python.

Input

The first line contains n ($1 \le n \le 2 \cdot 10^5$).

The second line contains integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$). The given integers are sorted by non-decreasing.

The third line contains m ($1 \le m \le 2 \cdot 10^5$).

The fourth line contains integers $b_1, b_2, \ldots, b_m$ ($-10^9 \le b_j \le 10^9$).

Output

Print the sequence of m integers, the j-th of them should be equal to such greatest i ($1 \le i \le n$) that $a_i = b_j$. If there is no such i, print `-1`.

Example

standard input
8 3 4 4 4 6 7 10 10 11 100 8 7 4 3 4 3 10 6 -100 2

standard output
-1 -1 6 4 1 4 1 8 5 -1 -1

Problem C. Difference with Closest

$l \leq n < r$

$l \quad r$

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

You are given two arrays of integer numbers: a and b. For each element of b find minimal difference with some element of a. I.e. for each b_j find the minimal value of $|b_j - a_i|$ over all possible indices i.

Input

The first line contains integer n ($1 \leq n \leq 10^5$) — the length of a. The second line contains $a_1, a_2, \ldots, a_n$ — the elements of a. They are integer numbers between -10^9 and 10^9, inclusive.

The third line contains integer m ($1 \leq m \leq 10^5$) — the length of b. The fourth line contains $b_1, b_2, \ldots, b_m$ — the elements of b. They are integer numbers between -10^9 and 10^9, inclusive.

Output

Print m numbers, where the j-th number is the minimal value of $|b_j - a_i|$.

Example

standard input	standard output
6 6 2 6 3 1 10 4 1 7 20 8	0 1 10 2

1, 2, 3, 6, 6, 10
0 1 2 3 4 5

diff arr[n(a)];

for each i in b

arr n[i] = flag = bin search for i in a

if flag

Print flag

else

print min(upper bound − i, −lower bound + i)

Problem D. Threads

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

You are given n spools of threads. The 1-st has a_1 meters of a thread, the 2-nd has a_2 meters of a thread, ..., the last (the n-th) has a_n meters of a thread.

Beautiful embroidery requires k pieces of thread of equal length. It is required to find out what is the greatest length of each of the k equal pieces, if:

- threads can not be tied (can not be join), they can only be cut;
- Each of k pieces has a length equal to an integer number of meters.

Input

The first line contains two integers n and k ($1 \le n, k \le 1000$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100000$, for each $i = 1 \ldots n$).

Output

Print the only number — the greatest integer s that it is possible to cut k equal pieces of a thread, each of length s. If it is impossible to find such integer positive s, print 0.

Examples

standard input	standard output
2 5 2 4	1
5 9 5 2 5 7 20	3
5 12 10 20 22 9 40	7

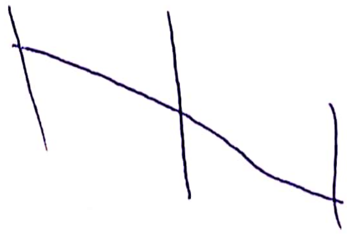
~~sort~~ arr[i] ← a_i

sort(arr);

~~for each k in (1, arr[len-1])~~

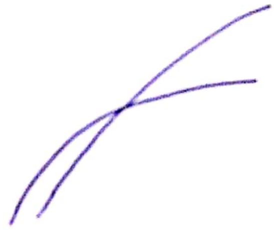
fun(k) = ~~ans~~ ~~map~~ reduce + (map $\frac{x}{k}$ arr)

do binary search on $\mathbb{R}$ (1, arr[len-1])



Problem E. Root of Tricky Equation

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes



Find a positive root of tricky equation for given positive C:

$$4x + 8\ln(x+1) + x \cdot \ln(x+1) = C.$$

Input

The input contains the only positive C ($0.001 \le C \le 100\,000$), it is given with at most 3 digits after decimal point.

Output

Print the root with at least 5 digits after the decimal point.

Examples

standard input	standard output
3.191	0.28252
1	0.08504

$4x + 8\ln(x+1) + x\ln(x+1) \le 100000$

$4x + \ln(x+1)(8+x) \le 100\,000.$

do 100 times

if $f(x) > 0$

mid = r

$x \le 7716.$

0

0 | 100 |

50

1, 2, 3

f(__) ______

0 10^4

run 100 times

guess = (l + r) / 2

if fun(guess) > 0

r = guess

else

l = guess

Problem F. Hamburgers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Polycarpus loves hamburgers very much. He especially adores the hamburgers he makes with his own hands. Polycarpus thinks that there are only three decent ingredients to make hamburgers from: a bread, sausage and cheese. He writes down the recipe of his favorite "Le Hamburger de Polycarpus" as a string of letters 'B' (bread), 'S' (sausage) and 'C' (cheese). The ingredients in the recipe go from bottom to top, for example, recipe "BSCBS" represents the hamburger where the ingredients go from bottom to top as bread, sausage, cheese, bread and sausage again.

Polycarpus has n_b pieces of bread, n_s pieces of sausage and n_c pieces of cheese in the kitchen. Besides, the shop nearby has all three ingredients, the prices are p_b rubles for a piece of bread, p_s for a piece of sausage and p_c for a piece of cheese.

Polycarpus has r rubles and he is ready to shop on them. What maximum number of hamburgers can he cook? You can assume that Polycarpus cannot break or slice any of the pieces of bread, sausage or cheese. Besides, the shop has an unlimited number of pieces of each ingredient.

Input

The first line of the input contains a non-empty string that describes the recipe of "Le Hamburger de Polycarpus". The length of the string doesn't exceed 100, the string contains only letters 'B' (uppercase English B), 'S' (uppercase English S) and 'C' (uppercase English C).

The second line contains three integers n_b, n_s, n_c ($1 \le n_b, n_s, n_c \le 100$) — the number of the pieces of bread, sausage and cheese on Polycarpus' kitchen. The third line contains three integers p_b, p_s, p_c ($1 \le p_b, p_s, p_c \le 100$) — the price of one piece of bread, sausage and cheese in the shop. Finally, the fourth line contains integer r ($1 \le r \le 10^{12}$) — the number of rubles Polycarpus has.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print the maximum number of hamburgers Polycarpus can make. If he can't make any hamburger, print 0.

Examples

standard input	standard output
BBBSSC 6 4 1 1 2 3 4	2
BBC 1 10 1 1 10 1 21	7
BSC 1 1 1 1 1 3 1000000000000	200000000001

$\frac{mb}{nb}$ $\frac{ms}{ns}$ $\frac{mc}{nc}$.

min div ( ) → can make..

$\Rightarrow f(k) = k(mb \cdot pb + ms \cdot ps + mc \cdot pc)$

$\cancel{B} =$

Problem G. K-th Element (Matrix Edition)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

You are given two integer sequences $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$. Consider $n \times m$ rectangular matrix c_{ij}, where $c_{i,j} = a_i + b_j$.

Sort all the elements of the matrix c_{ij}. There are $n \cdot m$ elements in total in the resulting sequence. Print the k-th element from this sequence.

Input

The first line contains three integers n, m, k ($1 \le n, m \le 45000$, $1 \le k \le n \cdot m$). The second line contains integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$). The third line contains integers $b_1, b_2, \ldots, b_m$ ($-10^9 \le b_j \le 10^9$).

Output

Print the k-th (1-base index) element of the sorted sequence of all elements of the matrix $c_{i,j} = a_i + b_j$.

Example

standard input	standard output
3 4 5 3 2 -2 3 4 4 -1	2

6 7 7 2
5 6 6 ~~1~~
1 2 2 ~~-3~~

c[0] = a[0] + b[0]
c[1] = a[0] + b[1]

4

-3 1 1 1 2
2 2 5 6
6 6 7 7

a[0] = 3 + 3
a[1] = 3 + 4
a[2] = 3 + 4
a[3] = 3 + (-1)
a[1e3+10].

Problem H. Annuity Payment Scheme

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

Very often, banks give loans (credits) on the terms of the annuity payment scheme. What does it mean?

For example, let a loan be given in s rubles for m months under p percent (monthly). Annuity payment denotes loan repayment in equal amounts (for example, x rubles). After such a payment in m months will be the amount paid equals $m \cdot x$ rubles.

Each monthly payment is divided into two parts:

- The first part (let's call it a_i) is used to pay off p percent of current debt. Obviously, $a_i = s' \cdot p/100$, where s' is the current amount of debt ($s' = s$ at the time of the first payment).
- The second part $b_i = x - a_i$ decreases the amount of current debt (i.e., reduces s'). After m payments, the value s' should turn to 0.

We assume that the bank uses real (floating point) numbers in the calculations.

For example, if $s = 100, m = 2, p = 50$, then $x = 90$. In the first month, $a_1 = s' \cdot p/100 = s \cdot p/100 = 50$, and $b_1 = 90 - 50 = 40$. Thus, $s' = 100 - 40 = 60$ at the time of the second payment. The second payment: $a_2 = 60 \cdot 50/100 = 30$, then $b_2 = 90 - 30 = 60$ and the loan is paid completely.

Your task is to find x by given s, m and p. Note that the answer is always unique (i.e. there the only such value x exists).

Input

The input contains integers s, m and p ($1 \le s \le 10^6, 1 \le m \le 120, 0 \le p \le 100$).

Output

Print the required value x with at least 5 digits after the decimal point.

Examples

standard input	standard output
`100 2 50`	`90.000000000`

Problem I. The Meeting Place Cannot Be Changed

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

$\sum_i \frac{(x_i - guess)}{v_i}$

The main road in Bytecity is a straight line from south to north. Conveniently, there are coordinates measured in meters from the southernmost building in north direction.

At some points on the road there are n friends, and i-th of them is standing at the point x_i meters and can move with any speed no greater than v_i meters per second in any of the two directions along the road: south or north.

You are to compute the minimum time needed to gather all the n friends at some point on the road. Note that the point they meet at doesn't need to have integer coordinate.

Input

The first line contains single integer n ($2 \le n \le 60\,000$) — the number of friends.

The second line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^9$) — the current coordinates of the friends, in meters.

The third line contains n integers $v_1, v_2, \ldots, v_n$ ($1 \le v_i \le 10^9$) — the maximum speeds of the friends, in meters per second.

Output

Print the minimum time (in seconds) needed for all the n friends to meet at some point on the road.

Your answer will be considered correct, if its absolute or relative error isn't greater than 10^{-6}. Formally, let your answer be a, while jury's answer be b. Your answer will be considered correct if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$ holds.

Examples

standard input	standard output
3 7 1 3 1 2 1	2.000000000000
4 5 10 3 2 2 3 2 4	1.400000000000

Note

In the first sample, all friends can gather at the point 5 within 2 seconds. In order to achieve this, the first friend should go south all the time at his maximum speed, while the second and the third friends should go north at their maximum speeds.

Problem J. Bars

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Polycarp's workday lasts exactly n minutes. He loves chocolate bars and can eat one bar in one minute. Today Polycarp has k bars at the beginning of the workday.

In some minutes of the workday Polycarp has important things to do and in such minutes he is not able to eat a chocolate bar. In other minutes he can either eat or not eat one chocolate bar. It is guaranteed, that in the first and in the last minutes of the workday Polycarp has no important things to do and he will always eat bars in this minutes to gladden himself at the begining and at the end of the workday. Also it is guaranteed, that k is strictly greater than 1.

Your task is to determine such an order of eating chocolate bars that the maximum break time between eating bars is as minimum as possible.

Consider that Polycarp eats a bar in the minute x and the next bar in the minute y $(x < y)$. Then the break time is equal to $y - x - 1$ minutes. It is not necessary for Polycarp to eat all bars he has.

Input

The first line contains two integers n and k $(2 \le n \le 200\,000,\ 2 \le k \le n)$ — the length of the workday in minutes and the number of chocolate bars, which Polycarp has in the beginning of the workday.

The second line contains the string with length n consisting of zeros and ones. If the i-th symbol in the string equals to zero, Polycarp has no important things to do in the minute i and he can eat a chocolate bar. In the other case, Polycarp is busy in the minute i and can not eat a chocolate bar. It is guaranteed, that the first and the last characters of the string are equal to zero, and Polycarp always eats chocolate bars in these minutes.

Output

Print the minimum possible break in minutes between eating chocolate bars.

Examples

standard input	standard output
`3 3` `010`	`1`

`8 3`
`01010110`
12345678
	`3`

Note

In the first example Polycarp can not eat the chocolate bar in the second minute, so the time of the break equals to one minute.

In the second example Polycarp will eat bars in the minutes 1 and 8 anyway, also he needs to eat the chocolate bar in the minute 5, so that the time of the maximum break will be equal to 3 minutes.

5-1-1   8-5-1

8-3-1

Problem K. Warehouse Position

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

There are n offices of a company on the axis Ox. Each office is characterized by its position (coordinate) x_i and the importance t_i. It is required to choose such a point on a straight line in order to build a warehouse in it so that the sum of "penalties" relatively all the offices is the lowest. "Penalty" to the office depends on the distance from the office to the warehouse and the importance of the office. Let the distance be d, then the penalty is:

- d, if $t_i = 1$,
- $d \cdot \ln(d+1)$ if $t_i = 2$,
- $d \cdot \sqrt{d}$ if $t_i = 3$,
- d^2 if $t_i = 4$.

Find a point on the line with minimal sum of "penalties" to all the offices.

Input

The first line contains integer n ($1 \le n \le 10^4$).

The following n lines contain pairs of integers x_i ($-10^6 \le x_i \le 10^6$) and t_i ($1 \le t_i \le 4$).

Output

In the first line print the required minimum total penalty with at least 6 digits after the decimal point. In the second line print the required point for building a warehouse with at least 6 digits after the decimal point.

The answer will be counted as correct if its absolute or relative error doesn't exeed 10^{-6}.

Examples

standard input	standard output
5 -2 2 4 3 -5 1 10 4 5 1	48.2872408823 6.3010139154
4 4 2 1 3 2 4 3 5 4 $2+\ln 5+0+1 = 3+\ln 5$	3.6923411760 3.9884351248 2 3 3 4 4 5 5

$4-2$
$t\ 1 = 1$ } => d ×

$\cancel{f(x) = (x-x_i)\cdot t_i}$

$f(x) = \sum (x - x_i)\cdot t_i$

$i,\ i+1$

3 → 1
4 → 2
5 → 3
4 → 4

2 3 3 4 4 5 5
2 1 3 2 4 3 5

Problem L. Minimum Value of Polynomial

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given the polynomial $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ is given. Find its minimum value on the segment $[l, r]$ (i.e. minimum of $P(x)$ for x in range $l \le x \le r$).

Input

The first line contains three integers n ($1 \le n \le 8$, $-100 \le l \le r \le 100$) — the degree of the polynomial and the ends of the segment. The second line contains a sequence of integer coefficients $a_0, a_1, \ldots, a_n$ ($-5 \le a_i \le 5$, $a_n \ne 0$).

Output

Output the required minimum value with at least 6 digits after the decimal point.

Examples

standard input	standard output
2 -100 100 1 2 1	0.0000000000
4 -2 2 2 -3 1 -4 3	-1.0331807052

$$P'(x) = n a_n x^{n-1} + n-1 a_{n-1} x^{n-2} + \cdots a_1 = 0$$

x^{n-1}