Educational Codeforces Round 61 (Rated for Div. 2)

A. Regular Bracket Sequence

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

A string is called *bracket sequence* if it does not contain any characters other than "(" and ")". A bracket sequence is called *regular* if it it is possible to obtain correct arithmetic expression by inserting characters "+" and "1" into this sequence. For example, "", "(())" and "()()" are regular bracket sequences; "))" and ")((" are bracket sequences (but not regular ones), and "(a)" and "(1)+(1)" are not bracket sequences at all.

You have a number of strings; each string is a bracket sequence of length 2. So, overall you have cnt_1 strings "((", cnt_2 strings "()", cnt_3 strings ")(" and cnt_4 strings "))". You want to write all these strings in some order, one after another; after that, you will get a long bracket sequence of length $2(cnt_1 + cnt_2 + cnt_3 + cnt_4)$. You wonder: is it possible to choose some order of the strings you have such that you will get a regular bracket sequence? **Note that you may not remove any characters or strings, and you may not add anything either**.

Input

The input consists of four lines, i-th of them contains one integer cnt_i ($0 \le cnt_i \le 10^9$).

Output

Print one integer: 1 if it is possible to form a regular bracket sequence by choosing the correct order of the given strings, 0 otherwise.

Examples

input
3 1 4 3
output
1

input
0 0 0 0
output
1

input
1 2 3 4
output
0

Note

In the first example it is possible to construct a string "(())()(()((()()()))())", which is a regular bracket sequence.

In the second example it is possible to construct a string "", which is a regular bracket sequence.

B. Discounts

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You came to a local shop and want to buy some chocolate bars. There are n bars in the shop, i-th of them costs a_i coins (and you want to buy all of them).

You have m different coupons that allow you to buy chocolate bars. i-th coupon allows you to buy q_i chocolate bars while you have to pay only for the $q_i - 1$ most expensive ones (so, the cheapest bar of those q_i bars is for free).

You can use only one coupon; if you use coupon i, you have to choose q_i bars and buy them using the coupon, and buy all the remaining $n - q_i$ bars without any discounts.

To decide which coupon to choose, you want to know what will be the minimum total amount of money you have to pay if you use one of the coupons optimally.

Input
The first line contains one integer n ($2 \le n \le 3 \cdot 10^5$) — the number of chocolate bars in the shop.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^9$), where a_i is the cost of i-th chocolate bar.

The third line contains one integer m ($1 \le m \le n - 1$) — the number of coupons you have.

The fourth line contains m integers q_1, q_2, ..., q_m ($2 \le q_i \le n$), where q_i is the number of chocolate bars you have to buy using i-th coupon so that the least expensive of them will be for free. **All values of q_i are pairwise distinct**.

Output
Print m integers, i-th of them should be the minimum amount of money you have to pay if you buy q_i bars with i-th coupon, and all the remaining bars one by one for their full price.

Example

input
7 7 1 3 1 4 10 8 2 3 4
output
27 30

Note
Consider the first example.

If we use the first coupon, we may choose chocolate bars having indices 1, 6 and 7, and we pay 18 coins for them and 9 coins for all other bars.

If we use the second coupon, we may choose chocolate bars having indices 1, 5, 6 and 7, and we pay 25 coins for them and 5 coins for all other bars.

C. Painting the Fence

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

You have a long fence which consists of n sections. Unfortunately, it is not painted, so you decided to hire q painters to paint it. i-th painter will paint all sections x such that $l_i \le x \le r_i$.

Unfortunately, you are on a tight budget, so you may hire only $q - 2$ painters. Obviously, only painters you hire will do their work.

You want to maximize the number of painted sections if you choose $q - 2$ painters optimally. A section is considered painted if at least one painter paints it.

Input
The first line contains two integers n and q ($3 \le n, q \le 5000$) — the number of sections and the number of painters availible for hire, respectively.

Then q lines follow, each describing one of the painters: i-th line contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output
Print one integer — maximum number of painted sections if you hire $q - 2$ painters.

Examples

input
7 5 1 4 4 5 5 6 6 7 3 5
output
7

input
4 3 1 1 2 2 3 4

output
2

input
4 4 1 1 2 2 2 3 3 4

output
3

D. Stressful Training

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berland SU holds yet another training contest for its students today. n students came, each of them brought his laptop. However, it turned out that everyone has forgot their chargers!

Let students be numbered from 1 to n. Laptop of the i-th student has charge a_i at the beginning of the contest and it uses b_i of charge per minute (i.e. if the laptop has c charge at the beginning of some minute, it becomes $c - b_i$ charge at the beginning of the next minute). The whole contest lasts for k minutes.

Polycarp (the coach of Berland SU) decided to buy a **single** charger so that all the students would be able to successfully finish the contest. He buys the charger at the same moment the contest starts.

Polycarp can choose to buy the charger with any non-negative (zero or positive) integer power output. The power output is chosen before the purchase, it can't be changed afterwards. Let the chosen power output be x. **At the beginning of each minute** (from the minute contest starts to the last minute of the contest) he can plug the charger into any of the student's laptops and use it for some **integer** number of minutes. If the laptop is using b_i charge per minute then it will become $b_i - x$ per minute while the charger is plugged in. Negative power usage rate means that the laptop's charge is increasing. The charge of any laptop isn't limited, it can become infinitely large. The charger can be plugged in no more than one laptop at the same time.

The student successfully finishes the contest if the charge of his laptop never is below zero at the beginning of some minute (from the minute contest starts to the last minute of the contest, zero charge is allowed). The charge of the laptop of the minute the contest ends doesn't matter.

Help Polycarp to determine the minimal possible power output the charger should have so that all the students are able to successfully finish the contest. Also report if no such charger exists.

Input

The first line contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 2 \cdot 10^5$) — the number of students (and laptops, correspondigly) and the duration of the contest in minutes.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$) — the initial charge of each student's laptop.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^7$) — the power usage of each student's laptop.

Output

Print a single non-negative integer — the minimal possible power output the charger should have so that all the students are able to successfully finish the contest.

If no such charger exists, print `-1`.

Examples

input
2 4 3 2 4 2

output
5

input
1 5 4 2

output
1

input
1 6 4 2
output
2

input
2 2 2 10 3 15
output
-1

Note

Let's take a look at the state of laptops in the beginning of each minute on the first example with the charger of power 5:

1. charge: $[3, 2]$, plug the charger into laptop 1;
2. charge: $[3 - 4 + 5, 2 - 2] = [4, 0]$, plug the charger into laptop 2;
3. charge: $[4 - 4, 0 - 2 + 5] = [0, 3]$, plug the charger into laptop 1;
4. charge: $[0 - 4 + 5, 3 - 2] = [1, 1]$.

The contest ends after the fourth minute.

However, let's consider the charger of power 4:

1. charge: $[3, 2]$, plug the charger into laptop 1;
2. charge: $[3 - 4 + 4, 2 - 2] = [3, 0]$, plug the charger into laptop 2;
3. charge: $[3 - 4, 0 - 2 + 4] = [-1, 2]$, the first laptop has negative charge, thus, the first student doesn't finish the contest.

In the fourth example no matter how powerful the charger is, one of the students won't finish the contest.

E. Knapsack

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

You have a set of items, each having some integer weight not greater than 8. You denote that a subset of items is good if total weight of items in the subset does not exceed W.

You want to calculate the maximum possible weight of a good subset of items. Note that you have to consider the empty set and the original set when calculating the answer.

Input

The first line contains one integer W ($0 \le W \le 10^{18}$) — the maximum total weight of a good subset.

The second line denotes the set of items you have. It contains 8 integers cnt_1, cnt_2, ..., cnt_8 ($0 \le cnt_i \le 10^{16}$), where cnt_i is the number of items having weight i in the set.

Output

Print one integer — the maximum possible weight of a good subset of items.

Examples

input
10 1 2 3 4 5 6 7 8
output
10

input
0 0 0 0 0 0 0 0 0
output
0

input

3 0 4 1 0 0 9 8 3
output
3

F. Clear the String

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a string s of length n consisting of lowercase Latin letters. You may apply some operations to this string: in one operation you can delete some contiguous substring of this string, if all letters in the substring you delete are equal. For example, after deleting substring bbbb from string abbbbaccdd we get the string aaccdd.

Calculate the minimum number of operations to delete the whole string s.

Input
The first line contains one integer n ($1 \le n \le 500$) — the length of string s.

The second line contains the string s ($|s| = n$) consisting of lowercase Latin letters.

Output
Output a single integer — the minimal number of operation to delete string s.

Examples

input
5 abaca
output
3

input
8 abcddcba
output
4

G. Greedy Subsequences

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

For some array c, let's denote a *greedy subsequence* as a sequence of indices p_1, p_2, ..., p_l such that $1 \le p_1 < p_2 < \cdots < p_l \le |c|$, and for each $i \in [1, l-1]$, p_{i+1} is the minimum number such that $p_{i+1} > p_i$ and $c[p_{i+1}] > c[p_i]$.

You are given an array $a_1, a_2, \ldots, a_n$. For each its subsegment of length k, calculate the length of its longest greedy subsequence.

Input
The first line contains two integers n and k ($1 \le k \le n \le 10^6$) — the length of array a and the length of subsegments.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — array a.

Output
Print $n - k + 1$ integers — the maximum lengths of greedy subsequences of each subsegment having length k. The first number should correspond to subsegment $a[1..k]$, the second — to subsegment $a[2..k+1]$, and so on.

Examples

input
6 4 1 5 2 5 3 6
output
2 2 3

input
7 6 4 5 2 5 3 6 6

output
3 3

Note

In the first example:

- $[1, 5, 2, 5]$ — the longest greedy subsequences are $1, 2$ ($[c_1, c_2] = [1, 5]$) or $3, 4$ ($[c_3, c_4] = [2, 5]$).
- $[5, 2, 5, 3]$ — the sequence is $2, 3$ ($[c_2, c_3] = [2, 5]$).
- $[2, 5, 3, 6]$ — the sequence is $1, 2, 4$ ($[c_1, c_2, c_4] = [2, 5, 6]$).

In the second example:

- $[4, 5, 2, 5, 3, 6]$ — the longest greedy subsequences are $1, 2, 6$ ($[c_1, c_2, c_6] = [4, 5, 6]$) or $3, 4, 6$ ($[c_3, c_4, c_6] = [2, 5, 6]$).
- $[5, 2, 5, 3, 6, 6]$ — the subsequence is $2, 3, 5$ ($[c_2, c_3, c_5] = [2, 5, 6]$).