A. K-Periodic Array

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

This task will exclusively concentrate only on the arrays where all elements equal 1 and/or 2.

Array a is k-period if its length is divisible by k and there is such array b of length k, that a is represented by array b written exactly $\frac{n}{k}$ times consecutively. In other words, array a is k-periodic, if it has period of length k.

For example, any array is n-periodic, where n is the array length. Array $[2, 1, 2, 1, 2, 1]$ is at the same time 2-periodic and 6-periodic and array $[1, 2, 1, 1, 2, 1, 1, 2, 1]$ is at the same time 3-periodic and 9-periodic.

For the given array a, consisting only of numbers one and two, find the minimum number of elements to change to make the array k-periodic. If the array already is k-periodic, then the required value equals 0.

Input

The first line of the input contains a pair of integers n, k $(1 \le k \le n \le 100)$, where n is the length of the array and the value n is divisible by k. The second line contains the sequence of elements of the given array $a_1, a_2, ..., a_n$ $(1 \le a_i \le 2)$, a_i is the i-th element of the array.

Output

Print the minimum number of array elements we need to change to make the array k-periodic. If the array already is k-periodic, then print `0`.

Sample test(s)

input
```
6 2
2 1 2 2 2 1
```
output
```
1
```

input
```
8 4
1 1 2 1 1 1 2 1
```
output
```
0
```

input
```
9 3
2 1 1 1 2 1 1 1 2
```
output
```
3
```

Note

In the first sample it is enough to change the fourth element from `2` to `1`, then the array changes to $[2, 1, 2, 1, 2, 1]$.

In the second sample, the given array already is 4-periodic.

In the third sample it is enough to replace each occurrence of number two by number one. In this case the array will look as $[1, 1, 1, 1, 1, 1, 1, 1, 1]$ — this array is simultaneously 1-, 3- and 9-periodic.

B. Fox Dividing Cheese

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

Two little greedy bears have found two pieces of cheese in the forest of weight a and b grams, correspondingly. The bears are so greedy that they are ready to fight for the larger piece. That's where the fox comes in and starts the dialog: "Little bears, wait a little, I want to make your pieces equal" "Come off it fox, how are you going to do that?", the curious bears asked. "It's easy", said the fox. "If the mass of a certain piece is divisible by two, then I can eat exactly a half of the piece. If the mass of a certain piece is divisible by three, then I can eat exactly two-thirds, and if the mass is divisible by five, then I can eat four-fifths. I'll eat a little here and there and make the pieces equal".

The little bears realize that the fox's proposal contains a catch. But at the same time they realize that they can not make the two pieces equal themselves. So they agreed to her proposal, but on one condition: the fox should make the pieces equal as quickly as possible. Find the minimum number of operations the fox needs to make pieces equal.

Input

The first line contains two space-separated integers a and b $(1 \le a, b \le 10^9)$.

Output

If the fox is lying to the little bears and it is impossible to make the pieces equal, print `-1`. Otherwise, print the required minimum number of operations. If the pieces of the cheese are initially equal, the required number is 0.

Sample test(s)

input
```
15 20
```
output
```
3
```

input
```
14 8
```
output
```
-1
```

input
```
6 6
```
output
```
0
```

C. Hamburgers

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

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) и '`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`.

Sample test(s)

input
```
BBBSSC
6 4 1
1 2 3
4
```
output
```
2
```

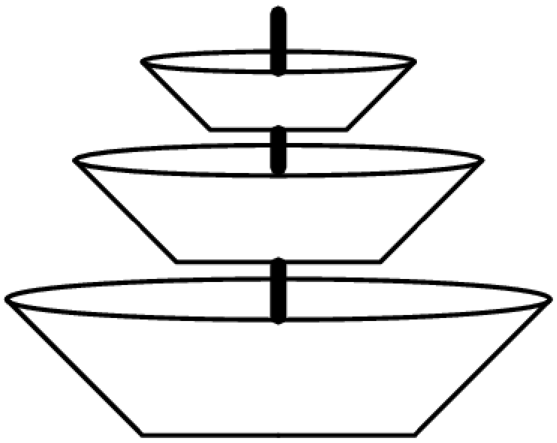
input
```
BBC
1 10 1
1 10 1
21
```
output
```
7
```

input
```
BSC
1 1 1
1 1 3
1000000000000
```
output
```
200000000001
```

D. Vessels

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

There is a system of n vessels arranged one above the other as shown in the figure below. Assume that the vessels are numbered from 1 to n, in the order from the highest to the lowest, the volume of the i-th vessel is a_i liters.



Initially, all the vessels are empty. In some vessels water is poured. All the water that overflows from the i-th vessel goes to the $(i+1)$-th one. The liquid that overflows from the n-th vessel spills on the floor.

Your task is to simulate pouring water into the vessels. To do this, you will need to handle two types of queries:

1. Add x_i liters of water to the p_i-th vessel;
2. Print the number of liters of water in the k_i-th vessel.

When you reply to the second request you can assume that all the water poured up to this point, has already overflown between the vessels.

Input

The first line contains integer n — the number of vessels $(1 \le n \le 2 \cdot 10^5)$. The second line contains n integers $a_1, a_2, ..., a_n$ — the vessels' capacities $(1 \le a_i \le 10^9)$. The vessels' capacities do not necessarily increase from the top vessels to the bottom ones (see the second sample). The third line contains integer m — the number of queries $(1 \le m \le 2 \cdot 10^5)$. Each of the next m lines contains the description of one query. The query of the first type is represented as "$1\ p_i\ x_i$", the query of the second type is represented as "$2\ k_i$" $(1 \le p_i \le n,\ 1 \le x_i \le 10^9,\ 1 \le k_i \le n)$.

Output

For each query, print on a single line the number of liters of water in the corresponding vessel.

Sample test(s)

input

```
2
5 10
6
1 1 4
2 1
1 2 5
1 1 4
2 1
2 2
```

output

```
4
5
8
```

input

```
3
5 10 8
6
1 1 12
2 2
1 1 6
1 3 2
2 2
2 3
```

output

```
7
10
5
```

E. Subway Innovation

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

Berland is going through tough times — the dirt price has dropped and that is a blow to the country's economy. Everybody knows that Berland is the top world dirt exporter!

The President of Berland was forced to leave only k of the currently existing n subway stations.

The subway stations are located on a straight line one after another, the trains consecutively visit the stations as they move. You can assume that the stations are on the Ox axis, the i-th station is at point with coordinate x_i. In such case the distance between stations i and j is calculated by a simple formula $|x_i - x_j|$.

Currently, the Ministry of Transport is choosing which stations to close and which ones to leave. Obviously, the residents of the capital won't be too enthusiastic about the innovation, so it was decided to show the best side to the people. The Ministry of Transport wants to choose such k stations that minimize the average commute time in the subway!

Assuming that the train speed is constant (it is a fixed value), the average commute time in the subway is calculated as the sum of pairwise distances between stations, divided by the number of pairs (that is $\frac{n\cdot(n-1)}{2}$) and divided by the speed of the train.

Help the Minister of Transport to solve this difficult problem. Write a program that, given the location of the stations selects such k stations that the average commute time in the subway is minimized.

Input

The first line of the input contains integer n $(3 \le n \le 3\cdot10^5)$ — the number of the stations before the innovation. The second line contains the coordinates of the stations $x_1, x_2, ..., x_n$ $(-10^8 \le x_i \le 10^8)$. The third line contains integer k $(2 \le k \le n - 1)$ — the number of stations after the innovation.

The station coordinates are distinct and not necessarily sorted.

Output

Print a sequence of k distinct integers $t_1, t_2, ..., t_k$ $(1 \le t_j \le n)$ — the numbers of the stations that should be left after the innovation in arbitrary order. Assume that the stations are numbered 1 through n in the order they are given in the input. The number of stations you print must have the minimum possible average commute time among all possible ways to choose k stations. If there are multiple such ways, you are allowed to print any of them.

Sample test(s)

input

```
3
1 100 101
2
```

output

```
2 3
```

Note

In the sample testcase the optimal answer is to destroy the first station (with $x = 1$). The average commute time will be equal to 1 in this way.