VK Cup 2016 - Round 3

A. Bear and Colors

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

Bear Limak has n colored balls, arranged in one long row. Balls are numbered 1 through n, from left to right. There are n possible colors, also numbered 1 through n. The i-th ball has color t_i.

For a fixed interval (set of consecutive elements) of balls we can define a *dominant* color. It's a color occurring the biggest number of times in the interval. In case of a tie between some colors, the one with the smallest number (index) is chosen as dominant.

There are non-empty intervals in total. For each color, your task is to count the number of intervals in which this color is dominant.

Input

The first line of the input contains a single integer n ($1 \le n \le 5000$) — the number of balls.

The second line contains n integers $t_1, t_2, ..., t_n$ ($1 \le t_i \le n$) where t_i is the color of the i-th ball.

Output

Print n integers. The i-th of them should be equal to the number of intervals where i is a dominant color.

Examples

input
4 1 2 1 2
output
7 3 0 0

input
3 1 1 1
output
6 0 0

Note

In the first sample, color 2 is dominant in three intervals:

- An interval $[2, 2]$ contains one ball. This ball's color is 2 so it's clearly a dominant color.
- An interval $[4, 4]$ contains one ball, with color 2 again.
- An interval $[2, 4]$ contains two balls of color 2 and one ball of color 1.

There are 7 more intervals and color 1 is dominant in all of them.

B. Bear and Two Paths

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

Bearland has n cities, numbered 1 through n. Cities are connected via bidirectional roads. Each road connects two distinct cities. No two roads connect the same pair of cities.

Bear Limak was once in a city a and he wanted to go to a city b. There was no direct connection so he decided to take a long walk, visiting each city **exactly once**. Formally:

- There is no road between a and b.
- There exists a sequence (path) of n distinct cities $v_1, v_2, ..., v_n$ that $v_1 = a$, $v_n = b$ and there is a road between v_i and v_{i+1} for .

On the other day, the similar thing happened. Limak wanted to travel between a city c and a city d. There is no road between them but there exists a sequence of n distinct cities $u_1, u_2, ..., u_n$ that $u_1 = c$, $u_n = d$ and there is a road between u_i and u_{i+1} for .

Also, Limak thinks that there are at most k roads in Bearland. He wonders whether he remembers everything correctly.

Given n, k and four distinct cities a, b, c, d, can you find possible paths $(v_1, ..., v_n)$ and $(u_1, ..., u_n)$ to satisfy all the given conditions? Find any solution or print `-1` if it's impossible.

Input

The first line of the input contains two integers n and k ($4 \le n \le 1000$, $n - 1 \le k \le 2n - 2$) — the number of cities and the maximum allowed number of roads, respectively.

The second line contains four **distinct** integers a, b, c and d ($1 \le a, b, c, d \le n$).

Output

Print `-1` if it's impossible to satisfy all the given conditions. Otherwise, print two lines with paths descriptions. The first of these two lines should contain n distinct integers $v_1, v_2, ..., v_n$ where $v_1 = a$ and $v_n = b$. The second line should contain n distinct integers $u_1, u_2, ..., u_n$ where $u_1 = c$ and $u_n = d$.

Two paths generate at most $2n - 2$ roads: $(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n), (u_1, u_2), (u_2, u_3), ..., (u_{n-1}, u_n)$. Your answer will be considered wrong if contains more than k distinct roads or any other condition breaks. Note that (x, y) and (y, x) are the same road.

Examples

input
7 11 2 4 7 3
output
2 7 1 3 6 5 4 7 1 5 4 6 2 3

input
1000 999 10 20 30 40
output
-1

Note

In the first sample test, there should be 7 cities and at most 11 roads. The provided sample solution generates 10 roads, as in the drawing. You can also see a simple path of length n between 2 and 4, and a path between 7 and 3.

C. Levels and Regions

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

Radewoosh is playing a computer game. There are n levels, numbered 1 through n. Levels are divided into k regions (groups). Each region contains some positive number of consecutive levels.

The game repeats the the following process:

1. If all regions are beaten then the game ends immediately. Otherwise, the system finds the first region with at least one non-beaten level. Let X denote this region.
2. The system creates an empty bag for tokens. Each token will represent one level and there may be many tokens representing the same level.
   - For each already beaten level i in the region X, the system adds t_i tokens to the bag (tokens representing the i-th level).
   - Let j denote the first non-beaten level in the region X. The system adds t_j tokens to the bag.
3. Finally, the system takes a uniformly random token from the bag and a player starts the level represented by the token. A player spends one hour and beats the level, even if he has already beaten it in the past.

Given n, k and values $t_1, t_2, ..., t_n$, your task is to split levels into regions. Each level must belong to exactly one region, and each region must contain non-empty consecutive set of levels. What is the minimum possible expected number of hours required to finish the game?

Input

The first line of the input contains two integers n and k ($1 \le n \le 200\,000$, $1 \le k \le min(50, n)$) — the number of levels and the number of regions, respectively.

The second line contains n integers $t_1, t_2, ..., t_n$ ($1 \le t_i \le 100\,000$).

Output

Print one real number — the minimum possible expected value of the number of hours spent to finish the game if levels are distributed between regions in the optimal way. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-4}.

Namely: let's assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct if .

Examples

input
4 2 100 3 5 7
output
5.7428571429

input
6 2 1 2 4 8 16 32
output
8.5000000000

Note

In the first sample, we are supposed to split 4 levels into 2 regions. It's optimal to create the first region with only one level (it must be the first level). Then, the second region must contain other three levels.

In the second sample, it's optimal to split levels into two regions with 3 levels each.

D. Bearish Fanpages

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a social website with n fanpages, numbered 1 through n. There are also n companies, and the i-th company owns the i-th fanpage.

Recently, the website created a feature called following. Each fanpage must choose exactly one other fanpage to follow.

The website doesn't allow a situation where i follows j and at the same time j follows i. Also, a fanpage can't follow itself.

Let's say that fanpage i follows some other fanpage j_0. Also, let's say that i is followed by k other fanpages $j_1, j_2, ..., j_k$. Then, when people visit fanpage i they see ads from $k+2$ distinct companies: $i, j_0, j_1, ..., j_k$. Exactly t_i people subscribe (like) the i-th fanpage, and each of them will click exactly one add. For each of $k+1$ companies $j_0, j_1, ..., j_k$, exactly people will click their ad. Remaining people will click an ad from company i (the owner of the fanpage).

The total income of the company is equal to the number of people who click ads from this copmany.

Limak and Radewoosh ask you for help. Initially, fanpage i follows fanpage f_i. Your task is to handle q queries of three types:

- `1 i j` — fanpage i follows fanpage j from now. It's guaranteed that i didn't follow j just before the query. Note an extra constraint for the number of queries of this type (below, in the Input section).
- `2 i` — print the total income of the i-th company.
- `3` — print two integers: the smallest income of one company and the biggest income of one company.

Input

The first line of the input contains two integers n and q ($3 \le n \le 100\,000$, $1 \le q \le 100\,000$) — the number of fanpages and the number of queries, respectively.

The second line contains n integers $t_1, t_2, ..., t_n$ ($1 \le t_i \le 10^{12}$) where t_i denotes the number of people subscribing the i-th fanpage.

The third line contains n integers $f_1, f_2, ..., f_n$ ($1 \le f_i \le n$). Initially, fanpage i follows fanpage f_i.

Then, q lines follow. The i-th of them describes the i-th query. The first number in the line is an integer $type_i$ ($1 \le type_i \le 3$) — the type of the query.

There will be at most $50\,000$ queries of the first type. There will be at least one query of the second or the third type (so, the output won't be empty).

It's guaranteed that at each moment a fanpage doesn't follow itself, and that no two fanpages follow each other.

Output

For each query of the second type print one integer in a separate line - the total income of the given company. For each query of the third type print two integers in a separate line - the minimum and the maximum total income, respectively.

Example

input

```
5 12
10 20 30 40 50
2 3 4 5 2
2 1
2 2
2 3
2 4
2 5
1 4 2
2 1
2 2
2 3
2 4
2 5
3
```

output

```
10
36
28
40
36
9
57
27
28
29
9 57
```

Note

In the sample test, there are 5 fanpages. The i-th of them has $i \cdot 10$ subscribers.

On drawings, numbers of subscribers are written in circles. An arrow from A to B means that A follows B.

The left drawing shows the initial situation. The first company gets income from its own fanpage, and gets income from the 2-nd fanpage. So, the total income is $5 + 5 = 10$. After the first query ("`2 1`") you should print 10.

The right drawing shows the situation after a query "`1 4 2`" (after which fanpage 4 follows fanpage 2). Then, the first company still gets income 5 from its own fanpage, but now it gets only from the 2-nd fanpage. So, the total income is $5 + 4 = 9$ now.

E. Bear and Destroying Subtrees

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Limak is a little grizzly bear. He will once attack Deerland but now he can only destroy trees in role-playing games. Limak starts with a tree with one vertex. The only vertex has index 1 and is a root of the tree.

Sometimes, a game chooses a subtree and allows Limak to attack it. When a subtree is attacked then each of its edges is destroyed with probability , independently of other edges. Then, Limak gets the penalty — an integer equal to the height of the subtree after the attack. The height is defined as the maximum number of edges on the path between the root of the subtree and any vertex in the subtree.

You must handle queries of two types.

- `1 v` denotes a query of the first type. A new vertex appears and its parent is v. A new vertex has the next available index (so, new vertices will be numbered $2, 3, ...$).
- `2 v` denotes a query of the second type. For a moment let's assume that the game allows Limak to attack a subtree rooted in v. Then, what would be the expected value of the penalty Limak gets after the attack?

In a query of the second type, Limak doesn't actually attack the subtree and thus the query doesn't affect next queries.

Input

The first line of the input contains one integer q $(1 \le q \le 500\,000)$ — the number of queries.

Then, q lines follow. The i-th of them contains two integers $type_i$ and v_i $(1 \le type_i \le 2)$. If $type_i = 1$ then v_i denotes a parent of a new vertex, while if $type_i = 2$ then you should print the answer for a subtree rooted in v_i.

It's guaranteed that there will be at least 1 query of the second type, that is, the output won't be empty.

It's guaranteed that just before the i-th query a vertex v_i already exists.

Output

For each query of the second type print one real number —the expected value of the penalty if Limak attacks the given subtree. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Namely: let's assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct if .

Examples

input
7 1 1 1 1 2 1 1 2 1 3 2 2 2 1
output
0.7500000000 0.5000000000 1.1875000000

input
8 2 1 1 1 1 2 1 3 1 4 2 1 1 4 2 1
output
0.0000000000 0.9375000000 0.9687500000

Note

Below, you can see the drawing for the first sample. Red circles denote queries of the second type.

F. Bears and Juice

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n bears in the inn and p places to sleep. Bears will party together for some number of nights (and days).

Bears love drinking juice. They don't like wine but they can't distinguish it from juice by taste or smell.

A bear doesn't sleep unless he drinks wine. A bear must go to sleep a few hours after drinking a wine. He will wake up many days after the party is over.

Radewoosh is the owner of the inn. He wants to put some number of barrels in front of bears. One barrel will contain wine and all other ones will contain juice. Radewoosh will challenge bears to find a barrel with wine.

Each night, the following happens in this exact order:

1. Each bear must choose a (maybe empty) set of barrels. The same barrel may be chosen by many bears.
2. Each bear drinks a glass from each barrel he chose.
3. All bears who drink wine go to sleep (exactly those bears who chose a barrel with wine). They will wake up many days after the party is over. If there are not enough places to sleep then bears lose immediately.

At the end, if it's sure where wine is and there is at least one awake bear then bears win (unless they have lost before because of the number of places to sleep).

Radewoosh wants to allow bears to win. He considers q scenarios. In the i-th scenario the party will last for i nights. Then, let R_i denote the maximum number of barrels for which bears surely win if they behave optimally. Let's define . Your task is to find , where denotes the exclusive or (also denoted as XOR).

Note that the same barrel may be chosen by many bears and all of them will go to sleep at once.

Input

The only line of the input contains three integers n, p and q $(1 \le n \le 10^9, 1 \le p \le 130, 1 \le q \le 2\,000\,000)$ — the number of bears, the number of places to sleep and the number of scenarios, respectively.

Output

Print one integer, equal to .

Examples

input
5 1 3
output
32

input
1 100 4
output
4

input
3 2 1
output
7

input
100 100 100
output
381863924

Note

In the first sample, there are 5 bears and only 1 place to sleep. We have $R_1 = 6, R_2 = 11, R_3 = 16$ so the answer is . Let's analyze the optimal strategy for scenario with 2 days. There are $R_2 = 11$ barrels and 10 of them contain juice.

- In the first night, the i-th bear chooses a barrel i only.
  - If one of the first 5 barrels contains wine then one bear goes to sleep. Then, bears win because they know where wine is and there is at least one awake bear.
  - But let's say none of the first 5 barrels contains wine. In the second night, the i-th bear chooses a barrel $5 + i$.

- If one of barrels $6 - 10$ contains wine then one bear goes to sleep. And again, bears win in such a situation.
- If nobody went to sleep then wine is in a barrel 11.

In the second sample, there is only one bear. He should choose an empty set of barrels in each night. Otherwise, he would maybe get wine and bears would lose (because there must be at least one awake bear). So, for any number of days we have $R_i = 1$. The answer is .

G. Choosing Ads

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

One social network developer recently suggested a new algorithm of choosing ads for users.

There are n slots which advertisers can buy. It is possible to buy a segment of consecutive slots at once. The more slots you own, the bigger are the chances your ad will be shown to users.

Every time it is needed to choose ads to show, some segment of slots is picked by a secret algorithm. Then some advertisers are chosen. The only restriction is that it should be guaranteed for advertisers which own at least $p\%$ of slots composing this segment that their ad will be shown.

From the other side, users don't like ads. So it was decided to show no more than ads at once. You are asked to develop a system to sell segments of slots and choose ads in accordance with the rules described above.

Input

The first line of the input contains three integers n, m and p ($1 \le n, m \le 150\,000, 20 \le p \le 100$) — the number of slots, the number of queries to your system and threshold for which display of the ad is guaranteed.

Next line contains n integers a_i ($1 \le a_i \le 150\,000$), where the i-th number means id of advertiser who currently owns the i-th slot.

Next m lines contain queries descriptions. Each description is of one of the following forms:

- `1 l r id` ($1 \le l \le r \le n, 1 \le id \le 150\,000$) — advertiser id bought all slots in a range from l to r inclusive;
- `2 l r` ($1 \le l \le r$) — you need to choose advertisers for segment $[l, r]$.

Output

For each query of the second type answer should be printed in a separate line. First integer of the answer should be the number of advertisements that will be shown . Next cnt integers should be advertisers' ids.

It is allowed to print one advertiser more than once, but each advertiser that owns at least slots of the segment from l to r should be in your answer.

Example

input

```
5 9 33
1 2 1 3 3
2 1 5
2 1 5
2 1 3
2 3 3
1 2 4 5
2 1 5
2 3 5
1 4 5 1
2 1 5
```

output

```
3 1 2 3
2 1 3
2 2 1
3 1 1000 1000
1 5
2 5 3
2 1 5
```

Note

Samples demonstrate that you actually have quite a lot of freedom in choosing advertisers.