Codeforces Round #351 (VK Cup 2016 Round 3, Div. 2 Edition)

A. Bear and Game

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

Bear Limak likes watching sports on TV. He is going to watch a game today. The game lasts 90 minutes and there are no breaks.

Each minute can be either interesting or boring. If 15 consecutive minutes are boring then Limak immediately turns TV off.

You know that there will be n interesting minutes $t_1, t_2, ..., t_n$. Your task is to calculate for how many minutes Limak will watch the game.

Input

The first line of the input contains one integer n $(1 \le n \le 90)$ — the number of interesting minutes.

The second line contains n integers $t_1, t_2, ..., t_n$ $(1 \le t_1 < t_2 < ... \ t_n \le 90)$, given in the increasing order.

Output

Print the number of minutes Limak will watch the game.

Examples

input

```
3
7 20 88
```

output

```
35
```

input

```
9
16 20 30 40 50 60 70 80 90
```

output

```
15
```

input

```
9
15 20 30 40 50 60 70 80 90
```

output

```
90
```

Note

In the first sample, minutes $21, 22, ..., 35$ are all boring and thus Limak will turn TV off immediately after the 35-th minute. So, he would watch the game for 35 minutes.

In the second sample, the first 15 minutes are boring.

In the third sample, there are no consecutive 15 boring minutes. So, Limak will watch the whole game.

B. Problems for Round

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

There are n problems prepared for the next Codeforces round. They are arranged in ascending order by their difficulty, and no two problems have the same difficulty. Moreover, there are m pairs of similar problems. Authors want to split problems between two division according to the following rules:

- Problemset of each division should be non-empty.
- Each problem should be used in exactly one division (yes, it is unusual requirement).
- Each problem used in division 1 should be harder than any problem used in division 2.
- If two problems are similar, they should be used in different divisions.

Your goal is count the number of ways to split problem between two divisions and satisfy all the rules. Two ways to split problems are considered to be different if there is at least one problem that belongs to division 1 in one of them and to division 2 in the other.

Note, that the relation of similarity **is not** transitive. That is, if problem i is similar to problem j and problem j is similar to problem k, it doesn't follow that i is similar to k.

Input

The first line of the input contains two integers n and m ($2 \le n \le 100\,000$, $0 \le m \le 100\,000$) — the number of problems prepared for the round and the number of pairs of similar problems, respectively.

Each of the following m lines contains a pair of similar problems u_i and v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$). It's guaranteed, that no pair of problems meets twice in the input.

Output

Print one integer — the number of ways to split problems in two divisions.

Examples

input

```
5 2
1 4
5 2
```

output

```
2
```

input

```
3 3
1 2
2 3
1 3
```

output

```
0
```

input

```
3 2
3 1
3 2
```

output

```
1
```

Note

In the first sample, problems 1 and 2 should be used in division 2, while problems 4 and 5 in division 1. Problem 3 may be used either in division 1 or in division 2.

In the second sample, all pairs of problems are similar and there is no way to split problem between two divisions without breaking any rules.

Third sample reminds you that the similarity relation is not transitive. Problem 3 is similar to both 1 and 2, but 1 is not similar to 2, so they may be used together.

C. 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.

D. 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.

E. 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.

F. 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.