VK Cup 2016 - Finals

A. LRU

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

While creating high loaded systems one should pay a special attention to caching. This problem will be about one of the most popular caching algorithms called LRU (Least Recently Used).

Suppose the cache may store no more than k objects. At the beginning of the workflow the cache is empty. When some object is queried we check if it is present in the cache and move it here if it's not. If there are more than k objects in the cache after this, the least recently used one should be removed. In other words, we remove the object that has the smallest time of the last query.

Consider there are n videos being stored on the server, all of the same size. Cache can store no more than k videos and caching algorithm described above is applied. We know that any time a user enters the server he pick the video i with probability p_i. The choice of the video is independent to any events before.

The goal of this problem is to count for each of the videos the probability it will be present in the cache after 10^{100} queries.

Input

The first line of the input contains two integers n and k $(1 \le k \le n \le 20)$ — the number of videos and the size of the cache respectively. Next line contains n real numbers p_i $(0 \le p_i \le 1)$, each of them is given with no more than two digits after decimal point.

It's guaranteed that the sum of all p_i is equal to 1.

Output

Print n real numbers, the i-th of them should be equal to the probability that the i-th video will be present in the cache after 10^{100} queries. You 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
3 1 0.3 0.2 0.5
output
0.3 0.2 0.5

input
2 1 0.0 1.0
output
0.0 1.0

input
3 2 0.3 0.2 0.5
output
0.675 0.4857142857142857 0.8392857142857143

input
3 3 0.2 0.3 0.5
output
1.0 1.0 1.0

B. Break Up

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

Again, there are hard times in Berland! Many towns have such tensions that even civil war is possible.

There are n towns in Reberland, some pairs of which connected by two-way roads. It is not guaranteed that it is possible to reach one town from any other town using these roads.

Towns s and t announce the final break of any relationship and intend to rule out the possibility of moving between them by the roads. Now possibly it is needed to close several roads so that moving from s to t using roads becomes impossible. Each town agrees to spend money on closing no more than one road, therefore, the total number of closed roads will be **no more than two**.

Help them find set of no more than two roads such that there will be no way between s and t after closing these roads. For each road the budget required for its closure was estimated. Among all sets find such that the total budget for the closure of a set of roads is minimum.

Input

The first line of the input contains two integers n and m ($2 \le n \le 1000$, $0 \le m \le 30\,000$) — the number of towns in Berland and the number of roads.

The second line contains integers s and t ($1 \le s, t \le n$, $s \ne t$) — indices of towns which break up the relationships.

Then follow m lines, each of them contains three integers x_i, y_i and w_i ($1 \le x_i, y_i \le n$, $1 \le w_i \le 10^9$) — indices of towns connected by the i-th road, and the budget on its closure.

All roads are bidirectional. It is allowed that the pair of towns is connected by more than one road. Roads that connect the city to itself are allowed.

Output

In the first line print the minimum budget required to break up the relations between s and t, if it is allowed to close no more than two roads.

In the second line print the value c ($0 \le c \le 2$) — the number of roads to be closed in the found solution.

In the third line print in any order c diverse integers from 1 to m — indices of closed roads. Consider that the roads are numbered from 1 to m in the order they appear in the input.

If it is impossible to make towns s and t disconnected by removing no more than 2 roads, the output should contain a single line `-1`.

If there are several possible answers, you may print any of them.

Examples

input

```
6 7
1 6
2 1 6
2 3 5
3 4 9
4 6 4
4 6 5
4 5 1
3 1 3
```

output

```
8
2
2 7
```

input

```
6 7
1 6
2 3 1
1 2 2
1 3 3
4 5 4
3 6 5
4 6 6
1 5 7
```

output

```
9
2
4 5
```

input

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

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

output

```
1
1
2
```

input

```
2 3
1 2
1 2 734458840
1 2 817380027
1 2 304764803
```

output

```
-1
```

C. Limak and Shooting Points

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

Bearland is a dangerous place. Limak can't travel on foot. Instead, he has k magic teleportation stones. Each stone can be used **at most once**. The i-th stone allows to teleport to a point (ax_i, ay_i). Limak can use stones **in any order**.

There are n monsters in Bearland. The i-th of them stands at (mx_i, my_i).

The given $k + n$ points are pairwise distinct.

After each teleportation, Limak can shoot an arrow in some direction. An arrow will hit the first monster in the chosen direction. Then, both an arrow and a monster disappear. It's dangerous to stay in one place for long, so Limak can shoot only one arrow from one place.

A monster should be afraid if it's possible that Limak will hit it. How many monsters should be afraid of Limak?

Input

The first line of the input contains two integers k and n ($1 \le k \le 7$, $1 \le n \le 1000$) — the number of stones and the number of monsters.

The i-th of following k lines contains two integers ax_i and ay_i ($-10^9 \le ax_i, ay_i \le 10^9$) — coordinates to which Limak can teleport using the i-th stone.

The i-th of last n lines contains two integers mx_i and my_i ($-10^9 \le mx_i, my_i \le 10^9$) — coordinates of the i-th monster.

The given $k + n$ points are pairwise distinct.

Output

Print the number of monsters which should be afraid of Limak.

Examples

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

input
3 8 10 20 0 0 20 40 300 600 30 60 170 340 50 100 28 56 90 180 -4 -8 -1 -2
output
5

Note

In the first sample, there are two stones and four monsters. Stones allow to teleport to points $(-2, -1)$ and $(4, 5)$, marked blue in the drawing below. Monsters are at $(4, 2)$, $(2, 1)$, $(4, -1)$ and $(1, -1)$, marked red. A monster at $(4, -1)$ shouldn't be afraid because it's impossible that Limak will hit it with an arrow. Other three monsters can be hit and thus the answer is 3.

In the second sample, five monsters should be afraid. Safe monsters are those at $(300, 600)$, $(170, 340)$ and $(90, 180)$.

D. Cron

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

Sometime the classic solution are not powerful enough and we have to design our own. For the purpose of this problem you have to implement the part of the system of task scheduling.

Each task should be executed at some particular moments of time. In our system you may set the exact value for the second, minute, hour, day of the week, day and month, when the task should be executed. Moreover, one can set a special value `-1` that means any value of this parameter is valid.

For example, if the parameter string is `-1 59 23 -1 -1 -1`, the problem will be executed every day at 23:59:00, 23:59:01, 23:59:02, ..., 23:59:59 (60 times in total).

Seconds, minutes and hours are numbered starting from zero, while day, months and days of the week are numbered starting from one. The first day of the week is Monday.

There is one special case that is treated separately. If both day of the week and day are given (i.e. differ from `-1`) to execute the task only one of these two (at least one, if both match this is fine too) parameters should match the current time (of course, all other parameters should match too). For example, the string of parameters `0 0 12 6 3 7` means that the task will be executed both on Saturday, July 2nd, 2016 and on Sunday, July 3rd, 2016 at noon.

One should not forget about the existence of the leap years. The year is leap if it's number is divisible by 400, or is not divisible by 100, but is divisible by 4. Each leap year has 366 days instead of usual 365, by extending February to 29 days rather than the common 28.

The current time is represented as the number of seconds passed after 00:00:00 January 1st, 1970 (Thursday).

You are given the string of six parameters, describing the moments of time the task should be executed. You are also given a number of moments of time. For each of them you have to find the first moment of time strictly greater than the current when the task will be executed.

Input

The first line of the input contains six integers s, m, h, day, $date$ and $month$ ($0 \le s, m \le 59$, $0 \le h \le 23$, $1 \le day \le 7$, $1 \le date \le 31$, $1 \le month \le 12$). Each of the number can also be equal to -1. It's guaranteed, that there are infinitely many moments of time when this task should be executed.

Next line contains the only integer n ($1 \le n \le 1000$) — the number of moments of time you have to solve the problem for. Each of the next n lines contains a single integer t_i ($0 \le t_i \le 10^{12}$).

Output

Print n lines, the i-th of them should contain the first moment of time strictly greater than t_i, when the task should be executed.

Examples

input

```
-1 59 23 -1 -1 -1
6
1467372658
1467417540
1467417541
1467417598
1467417599
1467417600
```

output

```
1467417540
1467417541
1467417542
1467417599
1467503940
1467503940
```

input

```
0 0 12 6 3 7
3
1467372658
1467460810
1467547200
```

output

```
1467460800
1467547200
1468065600
```

Note

The moment of time 1467372658 after the midnight of January 1st, 1970 is 11:30:58 July 1st, 2016.

E. Huffman Coding on Segment

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice wants to send an important message to Bob. Message $a = (a_1, ..., a_n)$ is a sequence of positive integers (*characters*).

To compress the message Alice wants to use binary Huffman coding. We recall that *binary Huffman code*, or *binary prefix code* is a function f, that maps each letter that appears in the string to some binary string (that is, string consisting of characters '0' and '1' only) such that for each pair of different characters a_i and a_j string $f(a_i)$ is not a prefix of $f(a_j)$ (and vice versa). The result of the encoding of the message $a_1, a_2, ..., a_n$ is the concatenation of the encoding of each character, that is the string $f(a_1)f(a_2)...f(a_n)$. Huffman codes are very useful, as the compressed message can be easily and uniquely decompressed, if the function f is given. Code is usually chosen in order to minimize the total length of the compressed message, i.e. the length of the string $f(a_1)f(a_2)...f(a_n)$.

Because of security issues Alice doesn't want to send the whole message. Instead, she picks some substrings of the message and wants to send them separately. For each of the given substrings $a_{l_i}...a_{r_i}$ she wants to know the minimum possible length of the Huffman coding. Help her solve this problem.

Input

The first line of the input contains the single integer n $(1 \le n \le 100\,000)$ — the length of the initial message. The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 100\,000)$ — characters of the message.

Next line contains the single integer q $(1 \le q \le 100\,000)$ — the number of queries.

Then follow q lines with queries descriptions. The i-th of these lines contains two integers l_i and r_i $(1 \le l_i \le r_i \le n)$ — the position of the left and right ends of the i-th substring respectively. Positions are numbered from 1. Substrings may overlap in any way. The same substring may appear in the input more than once.

Output

Print q lines. Each line should contain a single integer — the minimum possible length of the Huffman encoding of the substring $a_{l_i}...a_{r_i}$.

Example

input
7 1 2 1 3 1 2 1 5 1 7 1 3 3 5 2 4 4 4

output
10 3 3 5 0

Note

In the first query, one of the optimal ways to encode the substring is to map 1 to "0", 2 to "10" and 3 to "11".

Note that it is correct to map the letter to the empty substring (as in the fifth query from the sample).

F. Coprime Permutation

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

Two positive integers are coprime if and only if they don't have a common divisor greater than 1.

Some bear doesn't want to tell Radewoosh how to solve some algorithmic problem. So, Radewoosh is going to break into that bear's safe with solutions. To pass through the door, he must enter a permutation of numbers 1 through n. The door opens if and only if an entered permutation $p_1, p_2, ..., p_n$ satisfies:

In other words, two different elements are coprime if and only if their indices are coprime.

Some elements of a permutation may be already fixed. In how many ways can Radewoosh fill the remaining gaps so that the door will open? Print the answer modulo $10^9 + 7$.

Input

The first line of the input contains one integer n ($2 \le n \le 1\,000\,000$).

The second line contains n integers $p_1, p_2, ..., p_n$ ($0 \le p_i \le n$) where $p_i = 0$ means a gap to fill, and $p_i \ge 1$ means a fixed number.

It's guaranteed that if $i \neq j$ and $p_i, p_j \ge 1$ then $p_i \neq p_j$.

Output

Print the number of ways to fill the gaps modulo $10^9 + 7$ (i.e. modulo 1000000007).

Examples

input
4 0 0 0 0
output
4

input
5 0 0 1 2 0
output
2

input
6 0 0 1 2 0 0
output
0

input
5 5 3 4 2 1
output
0

Note

In the first sample test, none of four element is fixed. There are four permutations satisfying the given conditions: (1,2,3,4), (1,4,3,2), (3,2,1,4), (3,4,1,2).

In the second sample test, there must be $p_3 = 1$ and $p_4 = 2$. The two permutations satisfying the conditions are: (3,4,1,2,5), (5,4,1,2,3).

G. Cool Slogans

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Bomboslav set up a branding agency and now helps companies to create new logos and advertising slogans. In term of this problems, *slogan* of the company should be a non-empty substring of its name. For example, if the company name is "hornsandhoofs", then substrings "sand" and "hor" could be its slogans, while strings "e" and "hornss" can not.

Sometimes the company performs rebranding and changes its slogan. Slogan A is considered to be *cooler* than slogan B if B appears in A as a substring **at least twice** (this occurrences are allowed to overlap). For example, slogan $A =$ "abacaba" is cooler than slogan $B =$ "ba", slogan $A =$ "abcbcbe" is cooler than slogan $B =$ "bcb", but slogan $A =$ "aaaaaa" is not cooler than slogan $B =$ "aba".

You are given the company name w and your task is to help Bomboslav determine the length of the longest sequence of slogans $s_1, s_2, ..., s_k$, such that any slogan in the sequence is cooler than the previous one.

Input

The first line of the input contains a single integer n $(1 \le n \le 200\,000)$ — the length of the company name that asks Bomboslav to help. The second line contains the string w of length n, that consists of lowercase English letters.

Output

Print a single integer — the maximum possible length of the sequence of slogans of the company named w, such that any slogan in the sequence (except the first one) is cooler than the previous

Examples

input
3 abc
output
1

input
5 ddddd
output
5

input
11 abracadabra
output
3

Codeforces (c) Copyright 2010-2016 Mike Mirzayanov
The only programming contests Web 2.0 platform