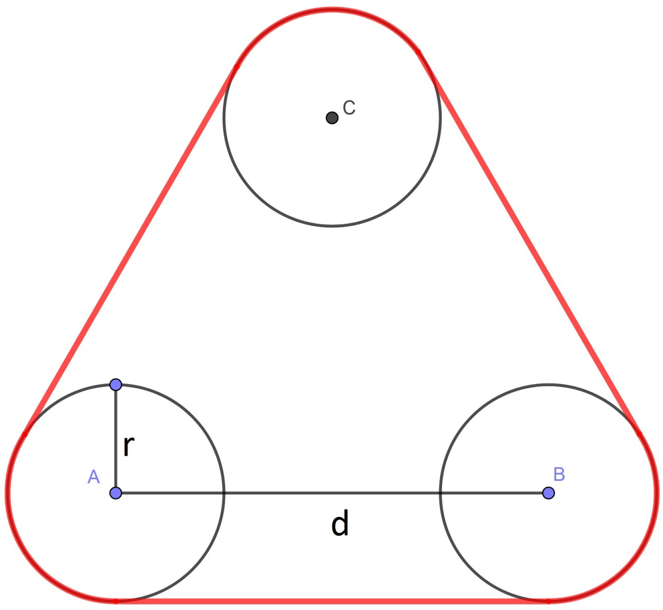
Problem A. Power Steering Belt

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The steering booster unit (SBU) in the Aysien's car stopped working because of the broken belt. This belt runs around three pulleys and transmits rotation from the drive pulley to the other two pulleys, one of which empowers the SBU. These pulleys are circles located equidistant from each other — the distance between any two is d cm (as shown in the diagram). The radius of each pulley is r cm.

Unfortunately, Aysien does not know what length belt to buy, as he bought the car recently. Help him to determine the length of the belt he needs.



Input

A single input line contains space separated two integers — d and r ($0 < d \le 1000$, $2 \cdot r \le d$).

Output

Output the length of the required belt with precision of two decimal places.

Scoring

There are no subtasks in this problem. Each test is evaluated independently of others.

Example

standard input	standard output
`30 10`	`152.83`

Note

The value of π should be assumed to be 3.14159265.

Problem B. Alien Meeting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

On a far far planet in the Tau Ceti system, intelligent beings have two hands with 11 fingers on each, so they use a 22-based numeral system. This system is also positional (like our usual decimal notation), but it uses following digits: $0, 1, 2, \ldots, 9, A, B, C, \ldots, J, K, L$.

During a recent meeting with taucetians, Erchim, who is a representative of the Earth, wondered what are the maximum and minimum n-digit taucetian numbers, composed of digits which do not exceed m, that are divisible by 2024.

Input

A single input line contains two space separated values: an integer decimal number n and a digit m in the taucetian numeral system ($2 \le n \le 10^6$, $m > 0$).

Output

Output two lines. First one should contain the maximum number with n digits in the taucetian numeral system. The second line — the minimum number.

If the problem has no as answer, then print out -1.

Scoring

Points for each test in each subtask are awarded independently of other tests and subtasks. The subtasks are evaluated independently of each other.

№	Additional constraints	Points	Feedback policy
1	$n \le 10$	20	complete
2	$11 \le n \le 1010$	30	complete
3	$n \ge 1011$	50	complete

Examples

standard input	standard output

`4 L`	`LIK0`
	`1320`

`5 9`	`99880`
	`10120`

Problem C. Energy Converter

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Humans have established a colony on Mars. Let's assume that the surface of Mars is a plane. Meteorites constantly fall on Mars. Humans have learned to alter the force with which the meteorites fall on the surface. The force of a meteorite impact is the radius of the shock wave caused by this event. The atmosphere of Mars has become very dense, so meteorites initially fall with a force of 0, i.e. without any shock wave.

Humanity is facing an energy crisis on Mars, so they have built energy converters, which generate 1 unit of energy if they are within the radius of a meteorite's shock wave. There are energy converters built at n different points on the surface of Mars.

The scientists have predicted the the next m meteorites. Their impact points all differ from each other.

The colonists have s batteries to charge the devices for altering the meteorite force. The device can increase the radius of the shock wave of any meteorite by r using up r batteries. Note that r is a positive integer and each battery can be used only once.

You are given the coordinates of the points where the energy converters are located and the coordinates of the points of m meteorites fall. Determine the maximum amount of energy that colonists can obtain using the devices for altering meteorite force.

Input

The first line contains two integers n and s ($1 \le n \le 10^4$, $0 \le s \le 1000$) — the number of energy converters and the number of batteries, respectively.

Each of the next n lines contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the points where the energy converters are located.

The next line contains an integer m ($1 \le m \le 100$) — the number of meteorites that will fall on Mars.

Each of the next m lines contains two integers u_i, v_i — the coordinates of a point where one of the meteorites will fall ($-10^9 \le u_i, v_i \le 10^9$).

It is guaranteed that all $n + m$ points are pairwise distinct.

Output

Output a single number — the maximum amount of energy that the colonists can obtain.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

№	Additional constraints	Points	Required subtasks	Feedback policy
1	$m = 2$	7	—	first failed test
2	$m \le 3$, $s \le 100$	8	—	first failed test
3	$n = 1$	12	—	first failed test
4	$n = 2$, $m \le 18$	13	—	first failed test
5	No additional constraints	60	S, 1, 2, 3, 4	first failed test

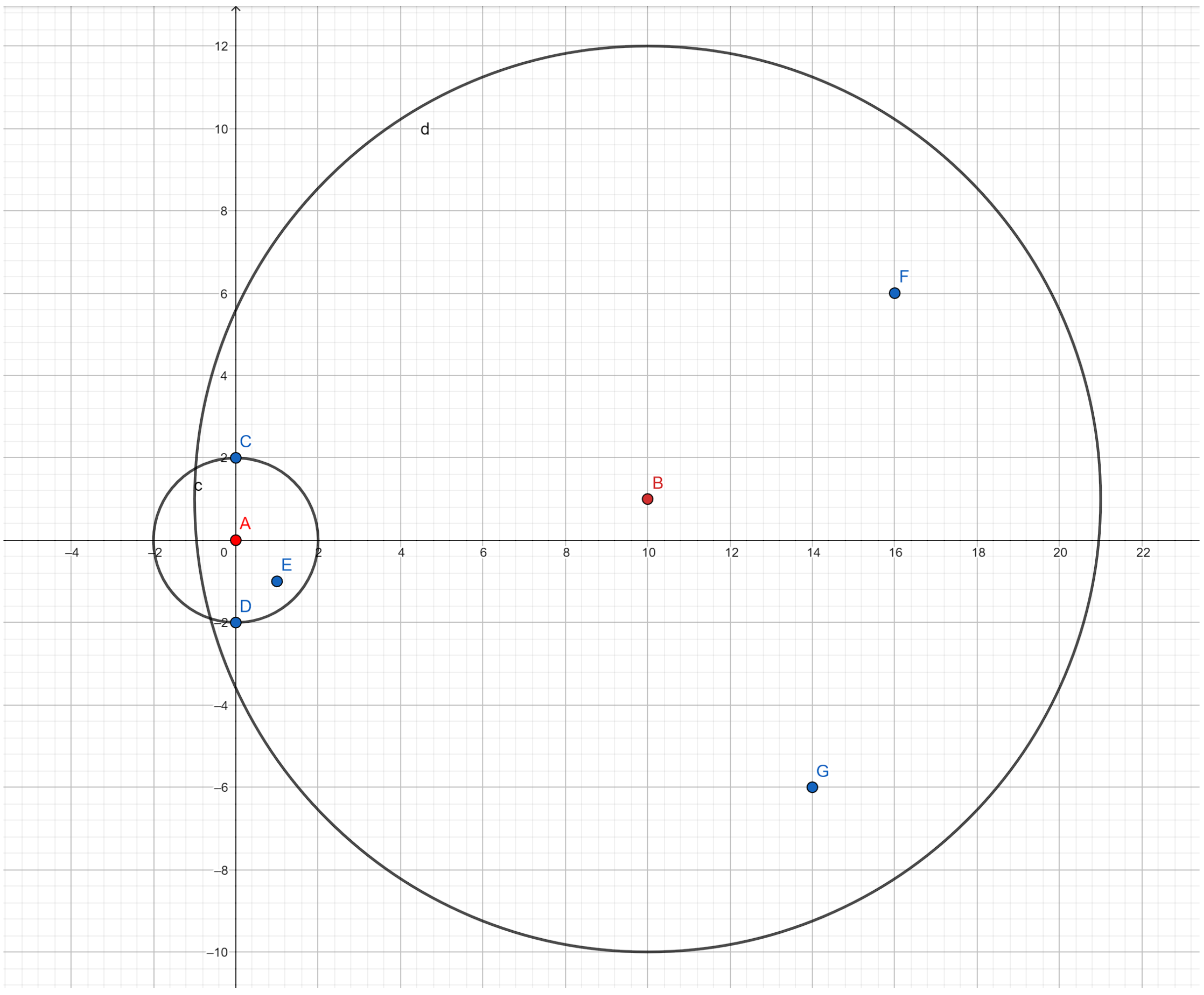
Example

standard input	standard output

`5 13`
`0 2`
`0 -2`
`1 -1`
`16 6`
`14 -6`
`2`
`0 0`
`10 1`
	`8`

Note

In the example given:



A and B are the points of meteorite impacts. The other points are the locations of energy converters. One of the ways to obtain the maximum number of energy units is to change the force of the first meteorite to 2, so that the shockwave will cover 3 converters, each of them will give 1 energy unit. The force of the second meteorite is changed to 11, covering all 5 converters. As a result, 8 energy units will be obtained.

Problem D. MultiChad

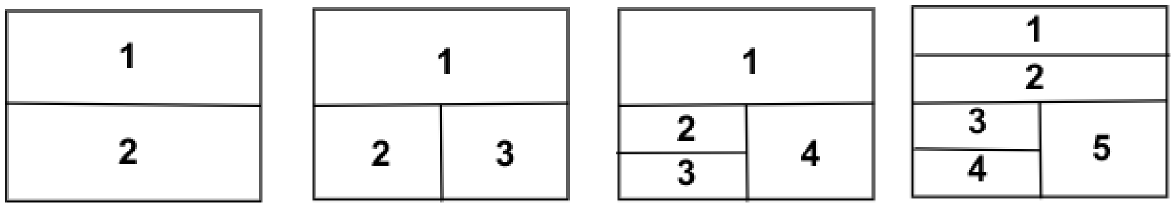
Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Kolya develops a new terminal multiplexer — “MultiChad”. Initially the terminal is a square window $[0, 2^k] \times [0, 2^k]$.

So far Kolya has developed support for only four commands. The first two of them divide the terminal window into subwindows:

- When the command `split v i` is received, the window number `i` is split by a vertical line into two identical subwindows. The left subwindow keeps its number `i`, and the right subwindow gets the number `i+1`. Accordingly, the numbering of all other windows starting from the (`i+1`)-th is incremented by 1. Also, in the future, when windows are split vertically, the numbers in the left half will always be smaller than the numbers in the right half.
- When the command `split h i` is given, the window number `i` is split horizontally into two identical subwindows. The upper half retains its number `i`, and the lower half gets the number `i+1`. Similarly, the numbering of all other windows starting from the (`i+1`)-th is incremented by 1. Also in the future, when windows are split horizontally, the numbers in the top half will always be smaller than the numbers in the bottom half.

For example, when the command sequence `split h 1`, `split v 2`, `split h 2`, `split h 1` is executed, the result looks like this:



The following two commands are also supported:

1. Command `where i` — the terminal reports the coordinates of the upper-left corner of the window numbered `i`.
2. Command `which y x` — the terminal reports which number window covers the point with coordinates `(y, x)`. If the point lies exactly on the window boundary, the window is considered to cover it. If the point is covered by two or more windows, the terminal reports the smallest of their numbers.

The top left point of the terminal has coordinates $(y = 0,\ x = 0)$, with the y-axis pointing down and the x-axis pointing to the right.

Input

The first line contains two integers: q — the number of commands and k (2^k — the length of the terminal side) $(0 < q \le 3 \cdot 10^5,\ 0 < k \le 62)$.

The following q lines specify commands in the described format. It is guaranteed that all commands are correct (commands like `split` and `where` take the number of the currently existing window, `split h` does not split a window of height 1, `split v` does not split a window of width 1, `which` takes integer coordinates) $(0 \le y, x \le 2^k)$.

Output

For each `where i` command, print on a separate line two numbers y and x separated by a space —the coordinates of the upper-left corner of window i.

For each `which y x` command, print on a separate line one number i — the number of the window containing the point (y, x).

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

№	**Additional constraints**	**Points**	**Required subtasks**	**Feedback policy**
1	$q \leq 140$, only commands `split v 1` and `where i`	10		first error
2	$q \leq 140$, only commands `split v 1`, `where i` and `which y x`	10	1	first error
3	Only commands `split v i` and `where i`	10	1	first error
4	Only commands `split v i`, `where i` and `which y x`	10	1, 2, 3	first error
5	Only commands `split v i`, `split h i` and `where i`	20	1, 2, 3	first error
6	No additional constraints	40	S, 1, 2, 3, 4, 5	first error

Examples

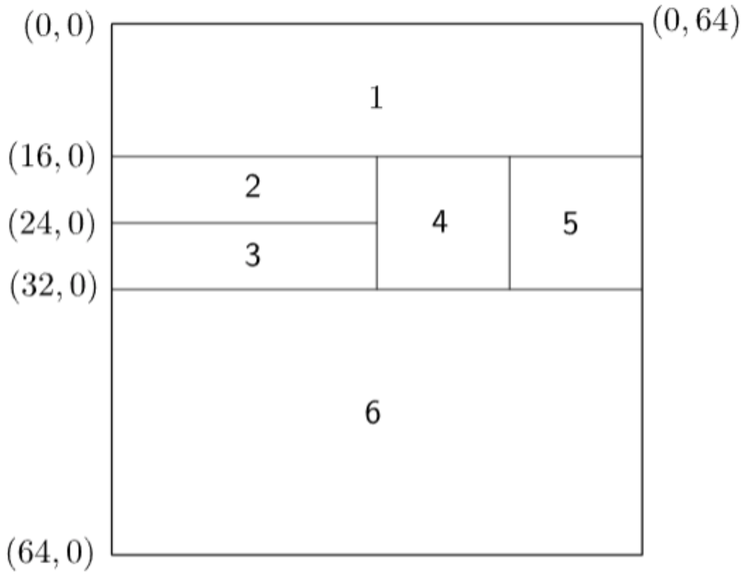
standard input	standard output

13 6
where 1
split h 1
split h 1
where 1
split v 2
split v 3
which 30 48
where 3
split h 2
where 5
where 6
which 24 5
which 10 60
	0 0
	0 0
	3
	16 32
	16 48
	32 0
	2
	1

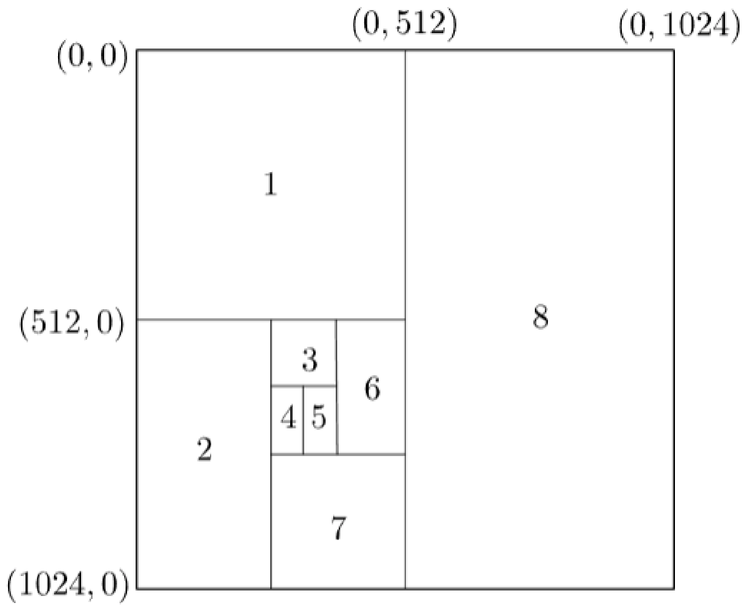
11 10
split v 1
split h 1
split v 2
which 400 256
split h 3
split v 3
where 6
split h 3
split v 4
which 256 768
where 5
	1
	0 512
	8
	640 320

Note

In the first example, the resulting terminal looks like this:



In the second example, the resulting terminal looks like this:



Problem E. The Bright Future

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Union X consists of N rapidly developing countries. There was a plan developed at the year 0, according to the plan each year a country with number A is chosen (each country has its unique number). The chosen country is invested 3^k coins (k is the year number). To ensure that the residents of other countries do not feel left out, countries that have received a total investment no greater than the country A, are also invested 3^k coins. Investments made before the year 0 are also taken into account.

New countries can join the Union X: when a country joins, it is considered to have received 0 coins of investments, even if it had been a member before. Countries can also leave the union. It is known that countries join and leave the union before choosing the country for investment.

You are given the data for K years that include the numbers of the selected countries and information about the countries joining and leaving the union X. List the countries in the union after those K years in a descending order of the total volume of investment.

Input

The first line contains two integers N and K ($1 \le N, K \le 10^5$).

In the next N lines, the numbers of the countries A_i and the previously invested amounts I_i are given ($0 \le A_i \le 10^9$, $0 \le I_i \le 10^9$).

Each of next K lines has the following format: the events of the k-th year are given: the count of countries J_k that joined the union followed by their numbers, the count of countries L_k that left the union followed by their numbers, as well as the number of the country chosen for investment A_k.

It is guaranteed that the sum of all J_k and all L_k does not exceed 10^5. Also, a country cannot leave and join the union in the same year. All numbers in each line are space separated.

Output

In the first line, print the count of countries after K years.

In the next line, print the numbers of the countries in the union in descending order of the total volume of investment after K years. If there are multiple correct answers, print any of them.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

№	Additional constraints	Points	Required subtasks	Feedback policy
1	$1 \le K \le 35$	10	S	first error
2	$N, K, \sum J_k, \sum L_k \le 10^3$	10	S	first error
3	$N, K, \sum J_k, \sum L_k \le 10^4$	40	S, 1, 2	first error
4	No additional constraints	40	S, 1, 2, 3	first error

Example

standard input	standard output

`3 3`
`1 4`
`2 10`
`3 9`
`1 4 0 3`
`0 0 1`
`0 1 4 1`
	`3`
	`1 3 2`

Note

In this example, in the year 0, first, the country with the number 4 joins the union X. Then, 1 coin ($3^0 = 1$) is invested into countries 1, 3, and 4.

In the beginning of year 1, the countries have the following total investments: country 1 — 5, country 2 — 10, country 3 — 10, and country 4 — 1. In this year, 3 coins ($3^1 = 3$) are invested in countries 1 and 4.

in the beginning of year 2, country 1 will have 8 coins invested, country 2 — 10, country 3 — 10, and country 4 — 4. In this year, country 4 leaves the union, so 9 coins ($3^2 = 9$) will be invested only into country 1.

After 3 years, a total of 17 coins will have been invested into country 1, and 10 coins each into countries 2 and 3.

Problem F. Teleportation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

A schoolboy Vasya is often late for school, and this has become such a big problem that he invented a teleporter in his garage.

Vasya's neighborhood can be represented as a rectangular matrix, the width of which is M, and the height is N. His house and school are located within the area. There are two types of cells in the matrix:

1. `.` — sidewalk or other cells where Vasya may walk. Let's say that such cells are *safe*.
2. `#` — puddles or roads that Vasya may not walk on. Let's say that such cells are *dangerous*.

Vasya can freely move one cell up, down, left or right on foot if there is a safe place there. That is, if he is in the cell (i, j), then he can go to the cell $(i + 1, j)$, $(i - 1, j)$, $(i, j + 1)$ or $(i, j - 1)$ if these cells are safe and within the matrix. This action does not spend any resources.

Upon the beginning, Vasya realized that the teleporter has a bug. It turns out that he teleports not to where Vasya wants, but to a point that is calculated as follows:

- The teleporter has a generator: $F(x, m) = (ax + c) \mod m$.
- If Vasya is in the cell with coordinates (i, j), then he will be teleported to $(F(i, N), F(j, M))$.

It is important to note that Vasya does not want to be in a dangerous cell, so teleportation should be carried out only when he is certain that he will be in a safe cell. Teleportation is available at any time from any safe cell.

The teleporter has K batteries. Each teleportation uses up 1 battery.

Your task is to determine if Vasya will be able to get to school, and if so, minimal amount of batteries necessary to do it?

Input

- The first line contains the positive integers N, M— the dimensions of the matrix ($N, M \leq 10^3$).
- The following N lines contain M characters each (either `"."` or `"#"`) describing the matrix.
- The next line contains two numbers — coordinates of the initial position of Vasya (i_0, j_0) ($0 \leq i_0 < N$, $0 \leq j_0 < M$).
- The next line contains two numbers — coordinates of the school (i_s, j_s) ($0 \leq i_s < N$, $0 \leq j_s < M$).
- The next line contains three positive integers a, c, K — generator parameters and the number of batteries in the teleporter ($a, c < 2^{32}$ and $K < 10^3$).

Output

Print the minimum number of batteries will be used up for Vasya to reach the school. If this is not possible, print -1.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

№	**Additional constraints**	**Points**	**Required subtasks**	**Feedback policy**
1	$N, M \leq 20$	20	S	first error
2	$N, M \leq 100$	30	S, 1	first error
3	No additional constraints	50	S, 1, 2	first error

Example

standard input	standard output

`6 4`
`#...`
`.###`
`....`
`.#.#`
`.#.#`
`#..#`
`0 1`
`2 0`
`7 3 14`
	`1`

Problem G. Alien Cipher

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In confrontation with the aliens, a partisan movement was created in Yakutia. During a secret mission, encrypted correspondence of the aliens was intercepted. This information can be extremely useful for the liberation of our planet, so it must be decrypted. Our spies have discovered the encryption algorithm.

As it turns out, the aliens are very strong physically, but their knowledge of cryptography is not yet as advanced as on Earth, and they have not yet mastered the algorithms of asymmetric encryption. Alien encryption works as follows: they take the secret information, encode it into binary form, then from the resulting binary string H of length N make an array A, where A_i $(0 \le i \le N)$ is the length of the longest palindrome substring of H starting from the i-th symbol. Actually, the intercepted information is an array A.

Your task is to recover string H from array A. Among all possible answers satisfying the given conditions, you need to find the lexicographically smallest one.

Input

The first line contains one integer N — the length of the string H $(1 \le N \le 10^5)$.

The second line contains N integers $A_1, A_2, \ldots, A_N$ $(1 \le A_i \le N)$ — the length of the longest palindrome in the string H, which starts with the i-th character.

Output

On a single line print the binary string H without spaces.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

№	**Additional constraints**	**Points**	**Required subtasks**	**Feedback policy**
1	$1 \le N \le 10$	10	S	first error
2	$1 \le N \le 10^4$	40	S, 1	first error
3	No additional constraints	50	S, 1, 2	first error

Example

`standard input`	`standard output`

`4`
`3 3 1 1`
	`0101`

Problem H. Factorial division

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Timur loves dividing numbers. Today at school, he learned about the factorial of a natural number: the factorial of k denoted by $k!$ is the product $1 \cdot 2 \cdot 3 \cdot \ldots \cdot (k-1) \cdot k$. With this new knowledge Timur came up with the following activity for himself:

1. First, he chooses a prime number p.
2. Then he constructs the following sequence $\{a\}_n$: $a_0 = p$, $a_{i+1} = a_i + k$, where k is the maximum power of the number p that divides the factorial of a_i for $i > 0$.

Timur wonders: for which initial prime p a given number x appears in the sequence as early as possible?

Input

In a single line, a single integer x is given ($2 \le x \le 4 \cdot 10^7$).

Output

Output two numbers separated by a space — the smallest possible index of x in the sequence and the corresponding initial prime p. It can be shown that the answer always exists under the given constraints. If there are multiple possible answers, output any of them.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

№	**Additional constraints**	**Points**	**Required subtasks**	**Feedback policy**
1	$x \le 100$	10	S	complete
2	$x \le 1000$	10	S, 1	complete
3	$x \le 10^4$	20	S, 1, 2	complete
4	$x \le 10^6$	30	S, 1, 2, 3	complete
5	No additional constraints	30	S, 1, 2, 3, 4	complete

Examples

`standard input`	`standard output`
`4`	`1 3`
`7`	`0 7`
`10`	`3 7`

Note

In the first example, you can take the prime number $p = 3$: then $a_0 = 3$, and $a_1 = 3 + 1$, because $3! = 6$ is divisible only by 3^1.

In the third example, the number 10 can be obtained from the prime number 7: the factorial $7! = 5040$ is divisible only by 7^1, $8! = 40320$ is divisible only by 7^1, $9! = 362880$ is divisible only by 7^1.

Problem I. T nights at NEFU

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Talban is a poor student at North-Eastern Federal University, who has just lost his scholarship. He had to apply for a job as a security guard at his university to earn some money. Unfortunately, Talban has classes during the day, so he can only work at night, from 12 am to 6 am. After T nights, he will receive his first paycheck of 120 dollars!

It is not that easy, though. At night, hungry monsters roam NEFU and will eat any person left in the building. Luckily, Talban's office has lasers that he can turn on to prevent monsters from entering. When turned on, lasers consume 1 unit of energy per second. One night lasts S seconds (time goes differently in NEFU), so Talban's shift begins at second 0 and ends at second S.

On the i-th night, n_i monsters will hunt Talban. Dayaana, Talban's clairvoyant friend, found out that the j-th monster will come near his office at second L_j and leave at second R_j. All this time Talban must keep the lasers on, spending 1 unit of energy per second. If there are many monsters near the office, lasers will still consume 1 unit of energy per second. In addition to lasers, Talban has a stun gun, which he can use once per night. When he uses the stun gun, all the monsters standing near the office immediately leave (and they don't come back), and lasers can be turned off for some time. The stun gun doesn't spend energy.

Since Talban will have limited energy, he wants to know for each night the least amount of energy he needs to spend in order to survive until the end of his shift.

Input

The first line contains three integers T, S, and G ($1 \le T \le 10^5$, $1 \le S \le 10^9$, $0 \le G \le 6$, where $G = 0$ corresponds to example tests from the statement) — the number of nights, the length of shift, and the subtask number respectively.

Then follow the descriptions of the T nights.

The first line of each night contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of monsters on this night.

Each of the following n lines contains two integers L_j and R_j ($0 \le L_j < R_j \le S$) — the segment of time during which the j-th monster will be near the office.

It is guaranteed that the sum of n over all nights does not exceed $2 \cdot 10^5$.

Output

For each night, output the least amount of energy that Talban must spend, given that he can use a stun gun.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

Let N be the sum of n over all nights.

№	Additional constraints	Points	Required subtasks	Feedback policy
1	$N, S \leq 1000$	15	S	first error
2	$N \leq 1000$	15	S, 1	first error
3	$S = 10^9$, $L_1 = 5 \cdot 10^8$, $R_1 = S$ and $R_i \leq L_1$ for each $i \geq 2$	15	—	first error
4	There will never be more than one monster near the office	5	—	first error
5	$S \leq 10^5$	25	S, 1	first error
6	No additional constraints	25	S, 1, 2, 3, 4, 5	first error

Example

standard input
```
5 11 0
1
3 9
3
3 5
6 11
0 3
4
0 4
3 5
1 4
2 6
6
4 6
2 8
0 6
8 11
3 8
1 10
5
2 11
3 10
4 9
5 8
6 7
```

standard output
```
0
5
3
7
4
```

Note

On the first night, Talban must use the stun gun at second 3, as soon as the first monster approaches the office. Then Talban will not use lasers at all, and will spend 0 energy.

On the second night, it is advantageous for Talban to turn on the lasers at second 0, and turn them off at 5, so that the first and third monsters do not eat him, and then shock the second monster at second 6. Lasers spent 5 energy.

On the third night, it is better for Talban to turn on the lasers at second 0, and turn them off at second 3, and then at the same second 3 he should shock all the monsters at the same time. Thus, he will drive them all away, and the lasers will spend only 3 energy.

On the fourth night, Talban should turn on the lasers at second 0, and turn them off at second 4, and then at the same second 4, shock all monsters except the fourth, since the fourth monster is not yet near the office. Next, Talban must turn on the lasers again at second 8, and then do nothing until the end of the shift (until second 11) so that the fourth monster does not eat him. Total lasers used $4 + 3 = 7$ energy.

Problem J. Alien taxes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

In a parallel reality, Yakutia consists of n cities and m undirected roads. From each city you can reach any other by roads. Every city has its own *budget* — amount of money in the treasury.

Everything was fine in the republic until aliens took over the entire Earth, including Yakutia. For a very long time, the new alien government collected taxes from poor humans.

Of course, the residents of the republic didn't like the new situation, and they decided to fight back. Each city has its *resistance* level — how strong the local population is. *Resistance* levels of all cities are pairwise distinct.

Humans were able to drive away the unwanted guests only after q days. Each of these q days, one of two events occurred in Yakutia:

- Type 1 event: city number v increases its *budget* by x via diamond sales.
- Type 2 event: aliens send their agents, with a strength level of c and plan p to the city v to collect taxes from the people. Agents decrease *budget* of all reachable cities by p. Agents can only travel via roads, since otherwise fauna of Yakutia will eat them alive. Also, they can't visit cities with *resistance* level higher than c, because the residents of that city will wipe them out. If the city's *budget* is less than p, aliens get all the money, leaving the city with the *budget* of zero. It is guaranteed that *resistance* level of the starting city v is not higher than c.

Now, after a long war with alien invaders, Yakutia needs to count losses. You, as the minister of alien affairs, were tasked to calculate for each event of the second type how much of the *budget* went to the aliens that day.

Input

The first line contains three integers n, m, and G ($1 \le n \le 3 \cdot 10^5$, $0 \le m \le 6 \cdot 10^5$, $0 \le G \le 9$, where $G = 0$ corresponds to example tests from the statement) — the number of cities, number of roads, and the subtask number respectively.

The second line contains n integers a_i — initial *budget* of the i-th city ($0 \le a_i \le 10^9$).

The third line contains n integers b_i — *resistance* level of the i-th city ($1 \le b_i \le n$, each number occurs exactly once).

The following m lines contain two integers u_i and v_i ($1 \le u_i, v_i \le n$) — the cities that the i-th road connects. It is guaranteed that every city is connected. Loops and multiple edges are acceptable.

The next line contains a single integer q ($1 \le q \le 3 \cdot 10^5$).

Then, the i-th of the following q lines contains an integer t_i — event type of the i-th day.

- If $t_i = 1$, then the line contains two more integers v_i and x_i ($1 \le v_i \le n$, $0 \le x_i \le 10^9$) — the city, and its *budget* increment respectively. Note that after this event city's *budget* may become greater than 10^9.
- If $t_i = 2$, then the line contains three more integers v_i, c_i, and p_i ($1 \le v_i \le n$, $1 \le c_i \le n$, $0 \le p_i \le 10^9$) — starting city of agents, their strength level, and their plan respectively.

Output

For each event of the second type, output one integer — how much of the *budget* went to the aliens that day.

Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks. Some subtasks may also require that all tests in the statement are completed. For such subtasks, the letter S is additionally specified in the required subtasks section.

Subtask	**Additional constraints**		**Points**	**Required subtasks**	**Feedback policy**
	n, m, q	Comment			
1	$n, q \leq 1000$, $m \leq 2000$	—	10	S	first error
2	—	$c_i = n$ for each event of the second type	10	—	first error
3	—	$v_i = 1$ for each event of the second type	10	2	first error
4	—	It is guaranteed that agents won't try to get more *budget* than the city has	10	—	first error
5	$n = m + 1$	$u_i = i$ and $v_i = i + 1$ for each road	10	—	first error
6	$n = m + 1$	$u_i = 1$ and $v_i = i + 1$ for each road	10	—	first error
7	$n = m + 1$	—	10	$5 - 6$	first error
8	$n, q \leq 10^5$, $m \leq 2 \cdot 10^5$	—	15	S, 1	first error
9	—	—	15	S, $1 - 8$	first error

Examples

standard input	standard output

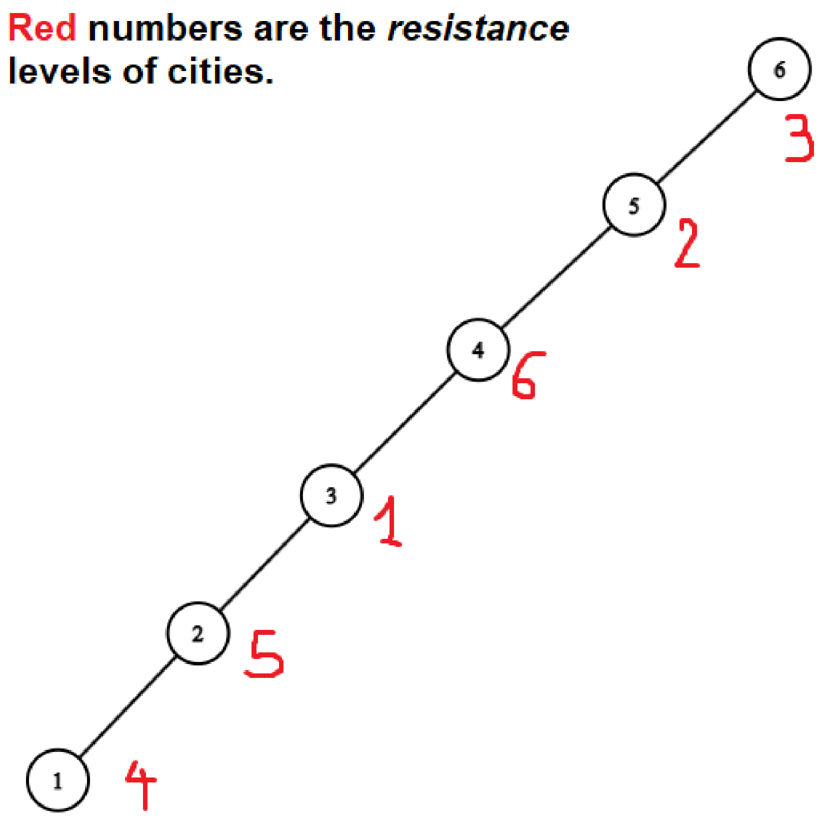
6 5 0
10 12 17 12 13 9
4 5 1 6 2 3
1 2
2 3
3 4
4 5
5 6
7
2 1 5 11
1 2 17
1 6 8
2 5 2 10
2 2 6 15
1 5 6
2 6 5 3
	32
	10
	51
	5

5 6 0
20 20 20 20 50
1 2 3 4 5
1 3
3 4
4 2
2 5
5 1
1 2
4
2 3 3 10
2 1 4 20
1 5 50
2 2 5 99
	30
	50
	99

1 0 0
15
1
3
1 1 20
2 1 1 5
2 1 1 40
	5
	30

Note

In the first example, Yakutia looks like this:



Initial *budgets* of cities: $10, 12, 17, 12, 13, 9$.

On the first day, aliens send agents with a plan of 11 and strength 5 to city 1. With a strength level of 5, they can reach cities 1, 2, and 3 because *resistance* of city 4 is too great. Agents collect 11 *budget* from cities 2 and 3, and 10 *budget* from city 1, since that's everything the city has. In total, agents collected $10 + 11 + 11 = 32$ *budget*.

Budgets after the first day: 0, 1, 6, 12, 13, 9.

On the second day, city 2 increases its *budget* by 17 via diamond sales.

Budgets after the second day: 0, 18, 6, 12, 13, 9.

On the third day, city 6 increases its *budget* by 8 via diamond sales.

Budgets after the third day: 0, 18, 6, 12, 13, 17.

On the fourth day, aliens send agents with a plan of 10 and strength 2 to city 5. This time, agents can't leave starting city 5 because *resistance* levels of adjacent cities are greater than the alien's strength, so they collect 10 *budget* from the only city 5.

Budgets after the fourth day: 0, 18, 6, 12, 3, 17.

On the fifth day, agents with a strength level of 6 and a plan of 15 start in city 2. Their strength allows them to visit every single city, thus, they collect $0 + 15 + 6 + 12 + 3 + 15 = 51$ *budget*.

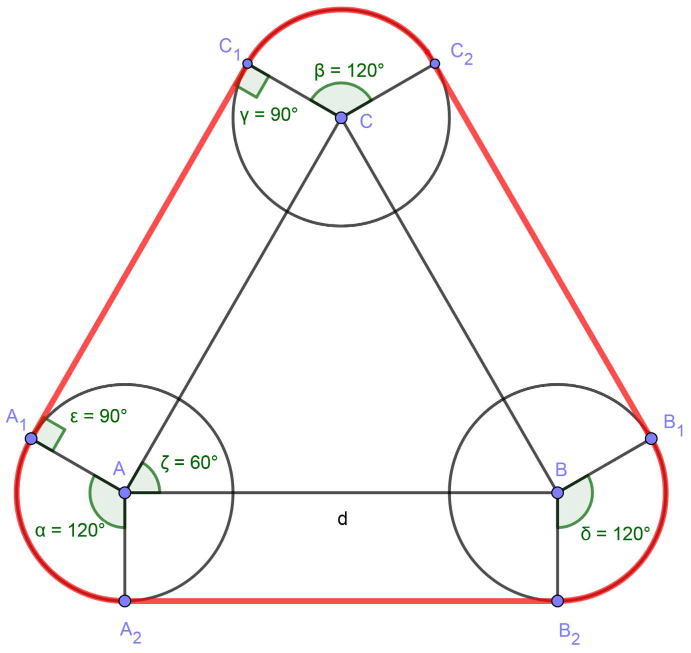
Budgets after the fifth day: 0, 3, 0, 0, 0, 2.

On the sixth day, city 5 increases its *budget* by 6.

Budgets after the sixth day: 0, 3, 0, 0, 6, 2.

On the last day, agents with a strength level of 5 and a plan of 3 start in city 6. They can reach cities 5 and 6, and in total, they collect $3 + 2 = 5$ *budget*.

Problem A. Power Steering Belt



The belt wraps around all three pulleys, hence the part of the belt connecting any two pulleys is tangent to them. Let's drop radii to the points of tangency. Without loss of generality, consider the quadrilateral AA_1C_1C. Then the angles $\angle AA_1C_1 = \angle CC_1A_1 = 90°$, and since $AA_1 = CC_1 = r$, it follows that $AC = A_1C_1 = d$. Similarly, $C_2B_1 = A_2B_2 = d$.

It remains to determine the lengths of the segments A_1A_2, B_1B_2, C_1C_2. Notice that $\angle A_1AA_2 = 360° - 60° - 90° - 90° = 120°$. Similarly, $\angle B_1BB_2 = \angle C_1CC_2 = 120°$, hence the sum of the lengths of the segments A_1A_2, B_1B_2, C_1C_2 is equal to the circumference of a circle with diameter r, i.e., $2\pi r$.

Therefore, the total length of the belt can be calculated using the formula $3d + 2\pi r$, which is what your program should output.

Problem B. Alien Meeting

To solve this problem, you can use several different algorithms.

While $n < 7$, you can use a brute force approach. However, this is an inefficient method. Since it's required to find numbers that are divisible by 2024, it will be faster, after finding the next k that is divisible by 2024, to immediately move to the number $k + 2024$. In this way, you can check numbers up to a length of $n < 11$.

However, to fully solve this problem, notice that $2024 = 8 \cdot 11 \cdot 23$, then let's use the divisibility rules for 8, 11, and 23.

The rule for divisibility by 8: the number formed by the last three digits must be divisible by 8. This rule works in the base-22 system in the same way as it does in the decimal system (the proof is based on the fact that the bases of both numeral systems are divisible by 2).

The rule for divisibility by 11 in base-22 numeral system: the last digit of the number must be 0 or B (11). This rule works similarly to the rule for divisibility by 5 in the decimal system. However, considering that the desired numbers must also be divisible by 8, it turns out that the last digit cannot be B. Therefore, using this rule means that the last digit of all desired numbers must be 0.

The rule for divisibility by 23 in base-22 numeral system: the difference between the sums of the digits in even and odd positions in an n-digit number must be divisible by 23. This rule is analogous to the rule for divisibility by 11 in the decimal system (to prove this, try multiplying the number by 23 by hand).

To solve the problem, let's note that in the largest desired number, the leftmost $n-5$ digits in the record will be the same. The specific digit can be easily selected in a loop. The remaining 5 digits on the right need to be iterated over to meet the divisibility rules.

The desired minimum number will always have 1 in the first position and a certain number $(n-6)$ of zeros. The remaining 5 digits on the right also need to be selected to meet the divisibility rules.

Problem C. Energy Converter

To solve the first subtask, it is enough to iterate over the radius of the first meteorite. Then calculate how much energy is generated in each of the possible options.

For the second subtask, a complete enumeration can also be used. We need to iterate over r_1 and r_2, the radii of the first and second meteorites respectively, and the radius of the third will be $s - r_1 - r_2$.

In the third subtask, each meteorite can cover at most one converter. We find the minimum radius for each meteorite at which its shock wave reaches the converter. Then we use a greedy algorithm and adjust those meteorites that require the fewest batteries to hit the converter.

In the fourth subtask, we will iterate over bit masks of length m. If the i-th bit of the mask is 1, then the i-th meteorite should cover two converters. Otherwise, it covers either 0 or 1 converter. Then we subtract from s the number of batteries spent on meteorites with a unit bit in the mask. Thus, we have s_1 batteries and some number of meteorites with a zero bit in the mask. To maximize the answer, we reduce the problem to the third subtask and use the greedy algorithm.

For the complete solution, we use dynamic programming. We calculate $dp[i][j]$ — the maximum amount of generated energy when using only the first i meteorites and j batteries. In the loop, we iterate i from 1 to m. We sort the points by distance from the i-th meteorite. Now, using two pointers, we can calculate $cnt[r]$ for all r from 1 to s — the number of converters that the i-th meteorite will cover if its radius is set to r. Knowing that $dp[i][j] = \max(dp[i-1][j], \max_{r=1\ldots j}(dp[i-1][j-r] + cnt[r]))$, we calculate $dp[i][j]$ for all j. In the end, the answer will be the maximum of $dp[m][j]$ for all j.

Problem D. MultiChad

1. In the first two subtasks, you only need to keep track of the number of executed commands `split v 1` — this number unambiguously defines the view of the entire terminal. This can be done in different ways. For example, you can honestly calculate the coordinates of the borders of all windows at each `where i` or `which y x` request and easily answer the request.

2. To solve the next two subtasks, a PBDS structure `ordered_set` can be used to support `split v i` commands (or a treap can be implemented, but that is overkill). When executing any `split` or `where` command, you must first find where window number `i` is — if you store the left x boundaries of all windows in ordered-set, then window i corresponds to the i-th element of the set. Then we respond to the `where` command immediately, and to the `split v i` command we need to create a new element for the new window (the right half of the current window i) and add it to the set. For the `which y x` command, we can binary search x to find the largest number i such that its left boundary does not exceed x — this is the answer.

3. To solve the last two subproblems, we should look at the problem in a different way. Let us represent the terminal as a binary tree — any window of the terminal corresponds to a vertex of the tree. When a window is divided into two subwindows — a vertex has two children. Also, for each vertex we will store the coordinates of its top left corner and its dimensions, as well as the number of leaves in its subtree. This allows us to find vertex i by descending the tree — if there is at least i on the left side of the tree, we descend to the left, otherwise to the right. The `where i` command can be answered immediately, and the `split` command creates two new vertices and hangs them from the current vertex, counting their coordinates and sizes (depends on the type of `split`). For `which y x` we also descend the tree — in a non-leaf vertex we know how the child windows are arranged and can calculate which one has the (y, x) point and descend to it.

Problem E. The Bright Future

Subtask 1

Notice that the number $a_i + \sum_{k=0}^{34} 3^k \le 10^9 + \frac{3^{35}-1}{2} < 2^{64}$ fits into a 64-bit integer data type. If we consider updating the coin counts each year in $O(n)$, adding and removing countries in $O(n)$, and finally sorting the array, we achieve an asymptotic complexity of $O(kn + n \log n)$.

Subtask 2

This subtask can still be solved by considering updates for both coin counts and countries each year in $O(n)$. However, since 3^{1000} doesn't fit into any primitive data type, we'll have to use arbitrary-precision arithmetic.

Subtask 3

First, calculate the first 20 years as in Subtask 1 and sort the cities in descending order. Then, maintain the countries in non-increasing order of investments. Note that $3^{20} > 10^9 + \sum_{k=0}^{19} 3^k$, so starting from the 20th year, countries that received investments will simply move to the beginning of the list, eliminating the need to explicitly calculate invested coins. This can be done using a regular array or deque, moving countries each year in $O(n)$. Adding and removing countries can be done in $O(n)$ time. The overall complexity is $O(20n + n \log n + kn)$.

Subtask 4

To improve the complexity of updates after the 20-th year, introduce an associative array for country numbers that can insert/delete/find elements in $O(\log n)$. Keep the count of invested coins in a linked list sorted, instead of country indices. Store pointers to linked list elements in the associative array to quickly find a country's position in the linked list (the pointer remains valid until deletion). As we can quickly find the cut-off point in the linked list, move its tail of countries that received investments to the front in $O(1)$ time. The final complexity is $O(20n + n \log n + k \log n)$. Using a hash-table we can achieve an average complexity of $O(20n + n \log n + k)$.

Problem F. Teleportation

First, you need to initialize the specified graph. Let's imagine points, i.e. safe coordinates, as vertices. Let's go through all the vertices and find all the vertices adjacent to them. Such vertices can be adjacent to 1 cell if they are safe, as well as a vertex to which Vasya can teleport if it is safe. If there is an edge between two vertices, its weight can be either 0 (if within walking distance) or 1 (if using a teleporter)

To solve the first subtask, you only need to use the Floyd-Warshall algorithm. Next, determine the distance between Vasya's starting position and the school. Let n be the number of vertices. The asymptotics of $O(n^3)$.

The second subproblem would require using Dijkstra's algorithm on a given graph, using a loop within a loop. The asymptotics of $O(n^2)$.

A complete solution would require using Dijkstra's algorithm, on a binary heap. For him, the asymptotics are $O(n \log n)$. But also, note that we can use the 0-1 BFS algorithm here, since the edges have a weight of either 0 or 1. For him, the asymptotics are $O(n)$.

Problem G. Alien Cipher

Let's set few statements, that will help us to get to the solution.

- As there can't be substring starting at i-th character longer $N - i + 1$ characters, we can assume that $1 \le A_i \le N - i + 1$ for any i ($1 \le i \le N$).
- For any $i < N$, if i-th and $i + 1$-th characters are same, there is always palindrome starting at i-th character with length at least 2. So $A_i > 1$, if $H_i = H_{i+1}$. We can also conclude that if $A_i = 1$, then $H_i \ne H_{i+1}$.

Having these statements, we can restore string H.

To solve this task we need to start from the last character. Let's assume its value as 1.

$$\underbrace{\_\,\_\,\_\,\cdots\,\_\,\_1}_{N\text{characters}}$$

For each j-th character starting from $N - 1$ to 1: if $A_j = 1$, then H_j is equal to opposite of $H_j + 1$, else $H_j = H_{j+A_j-1}$.

After applying this algorithm we will get some binary string H that will have proper set of characters to match the array A. But in the statement we have the task to find *lexicographically minimal* answer. To ensure this, we can check the first character of computed string H. If first character H_1 equals 1, we can just invert the whole string to get string H that will start with 0, that way we will get the lexicographically smaller string. And the resulting string will be our answer.

Problem H. Factorial division

Notice that any two consecutive prime numbers differ by at most a factor of 2 (such facts for given constraints could be verified by writing out the Sieve of Eratosthenes or by checking numbers for primality in the necessary range), and that the number $k!$ is divisible by p only once if $k < 2p$.

From this, it follows that the largest prime number p not exceeding the given number x defines a sequence where the number x will have an index $i = x - p$. Also, prime numbers in the range $\left(\frac{x}{2}; p\right)$ can be disregarded (since in them the number x will have an index exactly $i = x - q$ and $q < p$). Similarly, prime numbers greater than $x - p$ can be ignored, as the numbers in the sequence will grow by more than $x - p$ times exactly by 1. Since $x - p < 210$ within the given constraints, prime numbers need only be checked up to 210.

To quickly compute elements of the sequence, one can use the Legendre's formula (easily derived): the exponent of the prime number p in the factorial $k!$ is given by $v_p(k!) = \left\lfloor \frac{k}{p} \right\rfloor + \left\lfloor \frac{k}{p^2} \right\rfloor + \left\lfloor \frac{k}{p^3} \right\rfloor + \ldots.$

Problem I. T nights at NEFU

Let's reformulate the problem. We are given n segments. We can choose one point, and all the segments that contain the chosen point have their right boundary shifted to this point. We need to find the minimum possible length of the union of these segments.

Subtask 1

In this subtask, we can enumerate through all moments of time, at which we can use the stun gun. Then we shift the right bounds of the affected segments, and for $O(S)$ we consider the answer. The asymptotics of this solution is $O(S(S + n))$.

Subtask 2

Note that it is disadvantageous for us to use the stun gun when no new monsters appear at the selected time. Indeed, because we can use it a second earlier, and the answer will not be worse. Then we can enumerate through n left boundaries of the segments, and use the stun gun only in them. Then we shift the right boundaries of the affected segments, and we are left to count their union length. This well-known problem can be solved using scanline for $O(n \log n)$ (The logarithm appears from sorting the events). The final asymptotics is $O(n^2 \log n)$.

Subtask 3

In this subtask, it is advantageous for us to use the stun gun on the first monster to save $5 \cdot 10^8$ of energy. All other monsters attack only in the first half of the shift, and there is no more stun charge left on them. So the problem is reduced to the usual union of segments. Asymptotics is $O(n \log n)$.

Subtask 4

In this subtask, it is advantageous for us to use the stun gun on the monster that stands near the office the longest. Asymptotics is $O(n)$.

Subtask 5

Let's build a segment tree with mass operations on S elements, where i-th value is equal to the number of monsters standing outside the office at the i-th second (note that the right borders of the segments are not taken into account, because monsters leave at these moments). Initially add 1 on each segment, because we haven't used the stun gun yet.

Let's solve the problem by scanline. We will have two types of events: 1) monster arrives, and 2) monster leaves. Let's sort them in ascending order of coordinate (in case of equality, the events of the second type should go first). When we consider the first type event, we will say that we used the stun gun at that point in time. Then we will subtract 1 on the segment we are considering, since this monster left instantly. We also need to add a value equal to the number of monsters on the segment from the past event to the current event (this value can be easily maintained), because we moved the stun gun usage, and all the monsters on this segment have returned. When we consider the second type of event, we do the same thing, but we don't subtract 1 on the current segment, because the monster have left on its own.

Now we need to find out the answer for the current event. The answer is the number of non-zero elements in the entire array. To support this value, we will store the minimum and the number of minima in the ST node. The answer is S minus the number of minima on the whole segment, because the minimum is always zero. We end up with a solution for $O(n \log S + S)$.

Subtask 6

To get a full score on this problem, we need to compress the coordinates and get a solution for $O(n \log n)$.

Problem J. Alien taxes

Subtask 1

To solve this subtask, it is enough to use DFS to find all reachable vertices, and subtract budget from them. For each query of the second type, we answer for $O(n)$. The final asymptotics is $O(nq)$.

Subtask 4

Since it is guaranteed that agents will not take more budget than a vertex has, we are not interested in first type queries at all. For each query of the second type, we just need to find the number of reachable vertices to compute the answer. Let's answer the queries offline. For each possible strength of agents, let's keep all starting vertices. Suppose we initially have no vertices in the graph. Let's start a DSU, in which we maintain the sizes of the components. Let's go through all possible resistance levels from 1 to n. Suppose we are now considering resistance x. Let's add a vertex with this resistance to the graph, and use the DSU to merge it with all its neighbors. We then look at all the starting vertices of the agents with strength x, and find out the sizes of the components of there vertices. The DSU runs for $O(n\alpha(n))$. We find answers to queries for $O(1)$. The final asymptotics is $O(n\alpha(n) + q)$.

Subtask 2

In this subtask, agents can reach all vertices. Then, we need to support 3 types of queries: increase on a single element, subtract on the whole array, and for each element a_i do $a_i = max(a_i, 0)$. The Segment Tree Beats data structure handles all of this. However, there is a non-STB solution that uses a regular segment tree. Let's say that a vertex is zeroed, if its budget becomes zero. Then note that, in total, vertices will go from non-zero to zero at most $n + q$ times. Indeed, a zeroed vertex can become non-zero again after a first-type query, but there are at most q first-type queries. Then let us store in ST node the number of non-zero vertices, their total budget, the minimum budget, and the vertex with the minimum budget.

Now before reducing the budgets, let's find the number of non-zero vertices of cnt. The current answer is $cnt \times p$. Next, we reduce the budgets of all vertices. After this operation, some vertices may have been zeroed. Let's look at the vertex with minimal budget, if its budget is not positive, we subtract the extra budget from the current answer, mark this vertex as zeroed, and add $+1e18$ of budget to it, so that we don't consider it again. We repeat this process until the minimum becomes positive. At the end we will have the correct answer. The final asymptotics is $O((n + q) \log n)$.

Subtask 3

Same as in the second subtask, but we sort all vertices in advance in the order of increasing minimum possible resistance from vertex 1. Now we have all queries on the prefix. Asymptotics is $O((n+q) \log n+m)$.

Subtask 5

By binary search + sparse table for each query of the second type, we find to which vertex on the left and on the right the agents can reach. Now let's use the segment tree from the second subtask. Asymptotics is $O((n + q) \log n)$.

Subtask 6

Let's sort the vertices in the order of increasing resistance. Let's use the segment tree from the second subproblem. For each query of the second type, check if we can get to a vertex 1. If not, we reduce the budget of the starting vertex only. Otherwise, we will take away budgets from the prefix of vertices. Asymptotics $O((n + q) \log n)$.

Subtasks 7-9

In these subtasks, for each vertex we have to figure out which vertices the agents can reach, if their strength is equal to the resistance of this vertex. Let's build a tree. Let's start the DSU as in the fourth subtask, but instead of sizes we store the vertex with maximum resistance for each component. Let's go through the vertices in the order of increasing resistance. For the current vertex we consider all neighbors. If the resistance of a neighbor is greater than ours, or it is already in our component, we ignore it. Otherwise, we find the maximal vertex from the neighbor's component and add an edge in the tree between us and this maximal vertex. At the end we will have a tree with a root at the vertex with resistance n. Note that if agents start at some vertex with a strength equal to the resistance of that vertex, they can reach all vertices in the subtree, and only those vertices. However, the agents' strength is not necessarily equal to the starting resistance. In this case, it is possible to binary climb to the highest vertex, whose resistance is not greater than the agents' strength. Using Eulerian traversal, we reduce the tree problem to an array problem. Then we use the segment tree from the second subtask. The final asymptotics is $O((n + q) \log n + m)$. Depending on the optimality of the written solution, we get from 85 to 100 points.