Problem A

Multiset Machine

Time limit: 4.5 seconds
Memory limit: 256 megabytes

The mayor of Soteropolis, Betinho, which is known for his exotic tastes and for expending public money on unreasonable things, now decided to buy a Multiset Machine. A Multiset Machine can be represented by a DAG (directed acyclic graph) where each node represents an unary operation and each arc represents data flowing. The type of data that goes through an arc is a multiset of integers.

The incoming arcs of a node represent the input for such operation. If there are multiple incoming arcs, then the actual input for the operation is the sum of the multisets represented by those arcs.

The outgoing arcs of a node represent the output for such operation. The output is a single multiset of integers. If there are multiple outgoing arcs, then this multiset is replicated through all those arcs.

There are 4 types of nodes in a Multiset Machine. Some of them may have restrictions on the number of incoming/outgoing arcs. Also, they may have parameters which are properties of each node.

- `source` $v_1, \ldots, v_k$ – this node may have incoming arcs or not. If it has, then let I be the input multiset. Then this node outputs $I + \{v_1, \ldots, v_k\}$. Otherwise, it outputs $\{v_1, \ldots, v_k\}$.
- `print` – this node has no output. It prints the k-th smallest element of the input multiset. If there is no such element, then it prints `-1`. There is **exactly one** node of this type.
- `select` k_1, k_2 – this node has at least one incoming arc. It outputs a multiset comprised by a range of elements, from the k_1-th smallest element to the k_2-th smallest element of the input multiset. Every other element is ignored. This operation may produce less than $k_2 - k_1 + 1$ elements if for some $k_1 \leq i \leq k_2$ there is no i-th smallest element in the input multiset. In particular, if there is no k_1-th smallest element it will output an empty multiset.
- `generator` a, b, M – let I be the input for this node, or $\emptyset$ if there is no input. This node outputs $I + \{y \mid 0 \leq y < M \texttt{ and } \exists x, ax + b \equiv y \mod M\}$.

Every node with no incoming arc is necessarily a `source` or a `generator`, and every node with no outgoing arc is necessarily a `print`. When the machine is turned on, the data starts flowing from the nodes which have no input. A node will only process and output something when all of its inputs have been received. It is also guaranteed that the machine will eventually print something.

Your task is to write a program that, given the structure of the mayor's Multiset Machine, is capable of printing the same number that his machine would print if it was turned on.

Input

The first line contains two integers n, m ($2 \leq n \leq 10^5; 1 \leq m \leq 2 \times 10^5$) – the number of nodes and the number of arcs in the graph of the machine, respectively.

The following n lines contains the descriptions of the n nodes.

The i-th of these contains first a string representing the type of this node. If it is a `source` node, then there is a number k ($1 \leq k \leq 10^5$) followed by $v_1, \ldots, v_k$ on the same line. If it is a `select` node, then two integers k_1, k_2 ($1 \leq k_1, k_2 \leq 10^{18}$) follow. If it is a `generator`, then three integers a, b, M ($M > 0$) follow. If it is a `print`, then an integer k ($1 \leq k \leq 10^{18}$) follow.

Then the following m lines contains the arcs.

The i-th of them contains integers u_i, v_i ($1 \leq u_i, v_i \leq n; u_i \neq v_i$) meaning that there is an arc between u_i and v_i, in that order.

There may be parallel arcs, but not self-loops.

Every integer which has no constraints specified above are between 0 and 10^{18}, inclusive. The sum of k for every `source` does not exceed 10^5.

Output

Output the integer printed by the machine.

Samples

standard input	standard output
2 1 source 5 1 2 3 4 5 print 4 1 2	4

standard input	standard output
2 1 generator 2 1 7 print 3 1 2	2

standard input	standard output
4 3 generator 1 5 4 print 3 select 2 4 source 5 1 2 3 100 1000 1 3 4 3 3 2	2

standard input	standard output
3 2 source 2 4 5 select 1 1 print 2 1 2 2 3	-1

Problem B

Color Changing Sofa

Time limit: 1 second
Memory limit: 256 megabytes

Piva has recently bought a wonderful beach house at *Praia do Forte*.

After deciding that his house would be as colorful as a rainbow, Piva painted his living room in tiles of different colors. More specifically, Piva's living room can be seen as a row of n tiles such that the i-th of them has color s_i, where s is a string composed of lowercase English characters.

Piva now wants to put his brand-new *color changing* sofa in his room. The sofa is m ($m \leq n$) tiles long and should be put in the room in a way that it fits entirely on m consecutive tiles. Moreover, Piva wants the colors of the sofa and of the floor to match. We say that their colors match if the color of a sofa tile matches with the color of the floor tile **right below** it. To accomplish that, Piva may change the colors of the sofa according to its specification.

The sofa can be described by a string t of zeroes and ones of length m, where each character in the string represents a tile of the sofa. Piva can pick the zero-tiles and change them all to the same color. The same can be done for one-tiles. Therefore, notice that it is impossible for two zero-tiles (same for one-tiles) to have different colors. Also, that means that the tiles of the sofa will have **at most** two different colors. For instance, suppose the string is `101`. That means we can color the sofa `aba` or `aaa`, but not `abc`.

Given the descriptions of Piva's living room and of his sofa, determine on how many positions his sofa can be placed. Notice that for each possible placement, the sofa colors can be chosen independently. Also, notice that the sofa can be put in any direction (left to right **or** right to left), but a position where it can be put in both directions should be counted only once.

Input

The first line contains the string s ($|s| = n \leq 2000$) – the living room description.

The second line contains the string of zeroes and ones t ($|t| = m \leq n$) – the sofa description.

Output

Print a single integer – the number of positions the sofa can be placed.

Samples

standard input	standard output
aba 10	2

standard input	standard output
aba 111	0

standard input	standard output
aaa 101	1

standard input	standard output
abada 101	2

standard input	standard output
aab 100	1

Note

In the first example, we can put the sofa in two positions.

1. Position $[1, 2]$, color mapping $(0 \to a, 1 \to b)$, left to right.
2. Position $[2, 3]$, color mapping $(0 \to b, 1 \to a)$, left to right.

Notice there are other correct direction and color mapping configurations for each of these positions. The ones shown above are just examples.

In the second example, there is no way to place the sofa and to choose its colors such that there is a color match.

In the third example, we can put the sofa in $[1, 3]$, color mapping $(0 \to a, 1 \to a)$, left to right.

In the fourth example, we can put the sofa in two positions:

1. Position $[1, 3]$, color mapping $(0 \to a, 1 \to b)$, left to right.
2. Position $[3, 5]$, color mapping $(0 \to a, 1 \to d)$, left to right.

In the fifth example, we can put the sofa in $[1, 3]$, color mapping $(0 \to a, 1 \to b)$, **right to left**.

Problem C

Renan and Cirque du Soleil

Time limit: 1.5 seconds
Memory limit: 256 megabytes

Renan was driving the Cirque Du Soleil artists in his blue towner. Unfortunately the towner rolled over in Paralela (a well known avenue in Soteropolis). The N circus performers have each a distinct value between 1 and N. During the rollover the performers left the towner through the window, while doing some acrobatics and leaving poor Renan by himself inside the vehicle.

Everyone knows that Paralela is the habitat of an infamous Jaguar. This Jaguar wants to ambush the performers and get some food. Let S be the value of satisfaction of its lunch. S follows a totally arbitrary formula that can be calculated as shown below:

$$S = \sum_{T \subseteq P, T \neq \emptyset} |\max(T) - \min(T)|,$$

where P is the set of performers $\{1, 2, \ldots, N\}$, $\max(T)$ is the maximum value in T and $\min(T)$ is the minimum value in T.

Renan feels like he could be in trouble, so he wants to know the satisfaction the Jaguar will get by ambushing the performers. Since Renan is not good at maths, you need to calculate S for him. Note that this value can be big, so you only need to output it S module $10^9 + 7$.

Input

The first line contains an integer T ($1 \leq T \leq 10^4$), the number of testcases to be processed.

Each of the following T lines contains an integer N ($1 \leq N \leq 10^{18}$) – the number of performers Renan was driving with him.

Output

Output T lines, one for each testcase. In every line you need to print the value of S module $10^9 + 7$ for such testcase.

Sample

standard input	standard output
4 2 3 7 5	1 6 522 72

Problem D

Carnival

Time limit: 2.5 seconds
Memory limit: 256 megabytes

The Carnival is the biggest street festival in Soteropolis. It occurs in the very beginning of every Soteropolitan year and attracts a lot of people from around the globe. For the citizens of Soteropolis, though, it is actually a 50-50 situation: half of them love it, half of them hate it. That mainly occurs because it gets hard to transit in the city during the festival.

The mayor is planning to hold some tests before the actual Carnival. These tests are usually very small festivals which take place in a street of Soteropolis. During a festival, the street where it takes place gets blocked. Some of the streets of the city are called critical, but first we need to understand how the road system of Soteropolis was designed.

Soteropolis is composed by n junctions directly connected by m one-way streets. A street (u, v) means that a car can go from u to v but not the other way around. Also, Soteropolis was carefully designed such that from every junction it's possible to get to every other junction using the existent streets. A critical street is a street that, when blocked, makes it impossible to get from some junction u to some other junction v.

The mayor wants to hold a test festival. Your task is to decide which roads are critical and should be avoided.

Input

The first line contains two integers n, m ($2 \leq n \leq 2000; 1 \leq m \leq 50000$) – the number of junctions and the number of streets, respectively.

Then m lines follow. The i-th of them contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n; u_i \neq v_i$) – meaning that this is an one-way road between u_i and v_i. You may assume that the vertices are numbered from 1 to n.

There can be two or more streets between the same pair of junctions.

Output

In the first line output an integer k – the number of critical streets.

In the next k lines output the junctions connected by each of the streets. Each line should contain a pair of integers separated by space. Note those are ordered pairs since the streets are one-way, but the pairs themselves may be printed in any order.

Samples

standard input	standard output
3 3 1 2 2 3 3 1	3 1 2 2 3 3 1

standard input	standard output
4 6 1 2 2 3 3 4 4 1 2 4 3 1	3 1 2 2 3 4 1

Problem E

Hat-Xor

Time limit: 1 second
Memory limit: 256 megabytes

After having a wonderful time at the Carnival test festival, Jão and his friends decided to go for an after party in his house. At some point Jão convinced them to play a classical mathematical puzzle. He put his friends in a line and gave each one of them a typical Carnival hat colored in *black* or *white*. They were arranged in such a way that they could only see the hats of people that were in front of them. In particular, the last person could see everyone's hat and the first could see nothing.

This game must be played from the last one on the line to the first. Each person should try, in this order, to guess the color of his own hat **out loud**. For that, the participants can agree on some strategy **before** getting the hats.

Jão had so much fun watching his drunk friends trying to tackle the puzzle. Now he decided to teach them a powerful strategy. Let n be the number of participants in the game, and let they have numbers from 1 to n, from the first person in line to the last. Let the black hat have an associated value of 1 and the white one have a value of 0. Then we can always guess at least $n - 1$ values correctly by following this strategy:

1. The last one in line have no information about his own hat, so he should guess the `xor` ($\oplus$) of all the hats he can see;
2. The others players in line should do something different. Let X be the `xor` of the values guessed by previous players. Let Y be the `xor` of the hats the current player can see. If $X = Y$, then his own hat must be *white*, otherwise it must be *black*;

Notice that every player will follow (2), except for the last one in line, which will follow (1). This strategy ensures that the first $n - 1$ participants in line will guess correctly. Can you see why it works?

Unfortunately, Jão's friends are drunk, so they tend to mess the strategy up. When someone messes up, that means that he should have guessed x according to the strategy, but guessed $x \oplus 1$. Your task, given the guesses of each participant in line, is to decide if there is a hat distribution such that we can say no one messed up given those guesses.

Input

The only input line contains a binary string s ($|s| = n \leq 10^5$). The i-th character of s represents the color guessed by the i-th guy in the line (`0` for white, `1` for black).

Output

Output `YES` if there is a hat distribution such that we can say no one messed up. Otherwise, output `NO`.

Samples

standard input	standard output
10010	YES

standard input	standard output
01110	NO

Problem F

Renanzinho and His Toys

Time limit: 1 second
Memory limit: 256 megabytes

Renanzinho is a slow weirdo. In his 8th aniversary, there were candies, clowns, bouncers and even a delicious *strogonoff*. For him, the best part of the party, though, was unwrapping the gifts.

Renan, his father, gave to him N toys. The height of the i-th toy is given by A_i. Renanzinho loved the toys and is planning to spend the night rearranging them. He wants to arrange his toys into groups such that every group has a size between L and R and contains only contiguous toys (every group of toys is a **contiguous** subsequence of $1, 2, \ldots, N$). Also, every toy should be in **exactly** one group.

Suppose that Renanzinho splits his toys in K groups. The happiness of Renanzinho is arbitrarily defined as the minimum value among the maximum toy height of each group.

Since Renanzinho is a slow kid you need to calculate the maximum happiness he can achieve if he splits the toys optimally.

Input

The first line of the input contains three integers N, L, R ($1 \leq N \leq 10^5$;$1 \leq L \leq R \leq N$).

The second line of the input contains N integers. The i-th of them is A_i ($1 \leq A_i \leq 10^9$), the height of his i-th toy.

It's guaranteed that there is at least one way Renanzinho can split the toys.

Output

Output one line, the maximum value of happiness Renanzinho can achieve.

Samples

standard input	standard output
9 3 5 12 7 6 2 3 1 19 22 6	12

standard input	standard output
7 2 6 30 3 29 2 28 1 27	29

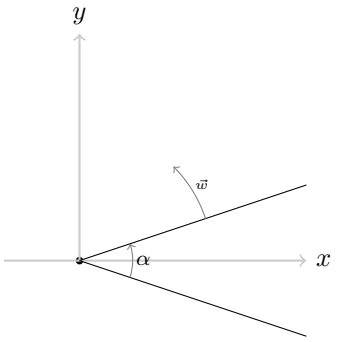
Problem G

Barra Lighthouse

Time limit: 1.5 seconds
Memory limit: 256 megabytes

The Barra Lighthouse is a famous touristic spot in Soteropolis. The lighthouse is 22 meters tall and its construction dates to 1698. At that time, the lighthouse was constructed right after the tragic wreck of the lead ship of the Commercial Soteropolis fleet.

It is known that the lamp of the lighthouse revolves in counter-clockwise direction with constant speed and a period of T seconds. Also, it is known that the emitted light is a beam with a spread angle of α degrees (see the picture below), and that the person responsible for monitoring the coast can only see ships that are **at most** d meters away from the lighthouse and that are **lightened**. We can imagine a system of two-dimensional coordinates with the lighthouse being on $(0, 0)$.



There are n ships in the coast of Soteropolis. Those ships are all at bearth, so their coordinates won't change. The i-th of them is at coordinates (x_i, y_i), $y_i > 0$. At the moment $t = 0$, the light beam has its center aligned with the x-axis positive direction, like in the picture shown above.

Your task is to compute, for each ship, for how much time it was visible to the guy in the lighthouse after X seconds have passed.

Input

The first line contains two integers n, X ($1 \leq n \leq 10^5$; $1 \leq X \leq 10^7$) – the number of ships in the coast and how many seconds have passed, respectively.

The second line contains two real numbers and one integer α, T, d ($0.5 \leq \alpha \leq 45.0$; $10^{-4} \leq T \leq 10^4$; $1 \leq d \leq 10^7$) – the spread angle of the light beam, the period of the lamp and the maximum distance a ship can be seen from the lighthouse, respectively.

Then n lines follow, each one with two integers describing x-coordinate and y-coordinate of a ship, respectively. The absolute value of the coordinates won't exceed 10^4 and y-coordinates will be strictly greater than zero.

The real numbers in the input will have no more than 5 decimal places.

Output

Output n lines. The i-th should have a real number – for how many seconds the i-th ship (in the order they appear in the input) was visible to the guy in the lighthouse, during the X seconds.

Your answer will be considered correct if its absolute error **or** its relative error does not exceed 10^{-4}.

Sample

standard input	standard output
3 5 15.0 2.2 5 1 1 1 5 -2 1	0.27500000000 0.00000000000 0.18333333333

Problem H

All-In

Time limit: 1 second
Memory limit: 256 megabytes

Concurrently to the 2018 ICPC Regionals, a huge international Poker event is being held in Soteropolis. Professional poker players from all around the world joined amateur Soteropolis players for a weekend of poker.

Two poker players in the event can join to play a head-to-head *all-in* game of poker. The organization of the event imposed a predetermined restriction n on the number of chips a player can have during a game. The number of chips the two players have can be denoted by a pair of integers (x, y). In a **valid** game, x, y should be between 1 and n. Also, $x + y$ shouldn't exceed n.

An *all-in* game of poker is a game where players will always bet $min(x, y)$. In each turn, the winner will get $min(x, y)$ chips and the loser will lose as many chips. If after a turn one of the players have no chips, the game is considered finished. Otherwise, the game will continue.

Unfortunately, there are scenarios where the game can last forever. For example, in a game where the players have $(2, 1)$, if the player with less chips always win, the game will never end. The organization of the event want to prohibit every game that **may possibly** last forever.

The games which will **for sure** end in a finite number of turns are called **finite**. Even finite games may take a long time to finish. Let $t(x, y)$ be the maximum time a finite game with chips (x, y) can last.

Your task is to compute the sum of $t(x, y)$ for every distinct (x, y) that represents a **valid finite** game, given the predetermined restriction n.

Notice that $(x, y) \neq (y, x)$ if $x \neq y$. Also notice that an initially valid game will remain valid, even if it lasts forever.

Input

The input contains T ($1 \leq T \leq 10^4$) testcases. The following T lines contains the descriptions of each testcase.

The only line of a testcase description contains an integer n ($2 \leq n \leq 10^{16}$) – the pretedermined restriction on the number of chips during a game.

Output

For each testcase, output a single integer in a line – the sum of $t(x, y)$ for every finite valid game.

It is guaranteed that these numbers will fit into a 64-bit signed integer.

Sample

standard input	standard output
4 2 3 4 7	1 1 6 7

Problem I

Colonial Mansions

Time limit: 2.5 seconds
Memory limit: 256 megabytes

After a year of hard work, Mano finally had a month off. He decided to take his grandmother to Soteropolis during this time. As soon as they arrived at the airport, Mano bought one of these "10 things to do in Soteropolis" travel guides. His grandmother instantly felt in love with pictures of the colonial mansions of Pelourinho, a historic neighborhood in western Soteropolis.

Pelourinho is a very big neighborhood. It is full of slopes and can be very trick for an elderly lady to explore. We can simplify its structure by imagining it as a line. The colonial mansions can be imagined as equally spaced spots on these lines: that is, from the i-th mansion (from left to right), you can go to mansions $i-1$ and $i+1$, if such mansions exist. These mansions are called neighbors of mansion i. Also, the i-th mansion is h_i meters above sea level.

Mano knows that his grandmother will have serious difficulties in this kind of terrain. Your task is to write a computer program to answer some queries and help Mano plan his trip to Pelourinho. The two types of queries are described below:

- `1` i H: the height of the i-th mansion should be set to H;
- `2` i H: how many mansions Mano and his grandmother can visit if they start their trip at mansion i and his grandmother can handle going from a mansion u to a neighbor mansion v iff $|h_u - h_v| \leq H$?

Write a program to process these queries.

Input

The first line of the input contains two integers n and q ($1 \leq n \leq 10^5$; $1 \leq q \leq 10^5$) - the number of colonial mansions and the number of queries you are asked to process, respectively.

The second line of the input contains n integers. The i-th of them is h_i ($0 \leq h_i \leq 10^9$) – the height above the sea level of the i-th mansion.

The next q lines of the input will each describe a query, which will have the format shown in the statement ($1 \leq i \leq n$; $0 \leq H \leq 10^9$).

Output

For each operation of type `2`, print a line with an integer – the number of colonial mansions Mano and his grandmother can visit in such scenario.

The queries should be processed in the order they are given in the input file.

Samples

standard input	standard output
4 3 1 2 1 3 2 2 1 1 4 1 2 3 1	3 4

standard input	standard output
4 5 1 10 2 11 2 3 1 2 4 1 1 3 8 2 2 3 2 1 9	1 1 3 4

Problem J

Soteros

Time limit: 3 seconds
Memory limit: 256 megabytes

Soteros is the biggest organized crime syndicate of Soteropolis. They are feared and respected. Members of *Soteros* follow a very rigid hierarchy. Sotero, the big boss, is the direct boss of some members, which in turn are direct boss of other members, and so on, like in a rooted tree structure. Therefore, Sotero is indirectly boss of the whole organization.

We can imagine a numbered rooted tree, where Sotero is represented by vertex 1. The other $n-1$ members are represented by numbers between 2 and n. The direct boss of the i-th member is p_i. Notice that every one has **exactly** one direct boss, except for Sotero.

Soteropolis is a city with K junctions and a bunch of two-way streets connecting pairs of these junctions. *Soteros* members have the culture of making themselves present on the streets. This is also true for the big boss Sotero. Every member of the organization is responsible for conducting business in **at most** one street of Soteropolis. More specifically, if the i-th member is responsible for some street, then junctions connected by this street are u_i, v_i.

Polo is an undercover agent working for SIA (Soteropolis Intelligence Agency) which finally got the chance to choose which member of *Soteros* he wants to work for. If he chooses to work for member i, then he will have free pass through all the streets that members (directly or indirectly) leadered by i are responsible for, but he won't be able to traverse *any* other streets of the town.

Polo definitely want to keep a low profile, but he won't be able to do much if he can't move around the city. He asked you to make an analysis to help on his decision.

A connected region of Soteropolis is a maximal set of junctions such that for every pair of these junctions there is a path between them consisting only of streets Polo can freely traverse. In particular, an isolated junction (with no streets Polo can traverse around it) is a connected region.

For each member of *Soteros*, you should compute the number of connected regions Polo will be able to traverse if he chooses to work for such member.

Input

The first line contains two integers n, K ($2 \leq n, K \leq 10^5$) – the number of members of *Soteros* and the number of junctions of Soteropolis, respectively.

The second line contains $n-1$ integers separated by spaces. The i-th of them is p_{i+1} ($1 \leq p_{i+1} \leq i$) – the direct boss of the $(i+1)$-th member.

The next n lines contains two integers each. If the i-th of these lines contains `0 0`, then the i-th member is responsible for no street. Otherwise, it contains two integers u_i, v_i ($1 \leq u_i, v_i \leq K$; $u_i \neq v_i$) – the streets which the i-th member is responsible for.

Two distinct members can be responsible for a street connecting the same pair of junctions u_i, v_i. That means they are responsible for the same street.

Output

Output n lines. The i-th of them should contain an integer – the number of connected regions Polo will be able to traverse if he chooses to work for member i.

Samples

standard input	standard output
3 3 1 2 0 0 1 2 2 3	1 1 2

standard input	standard output
8 7 1 1 1 2 2 5 5 1 2 2 3 0 0 6 7 2 7 7 5 7 3 5 2	2 4 7 6 4 6 6 6

Problem K

Rei do Cangaço

Time limit: 1 second
Memory limit: 256 megabytes

João has just bought a new game from the store: *Lampião, o Rei do Cangaço*. The main character of the game is Lampião, a known figure of Soteropolitan culture and probably the most famous *cangaceiro* of all time. There is a lot of controversy around the reputation of the *cangaceiros*, but João definitely must stick to their cause to beat this game.

In this game, João must control Lampião through a series of mini-games. There is one mini-game that caught João's attention: the *treasure hunt*. In this mini-game, there are n houses in a row, numbered 1 to n from left to right. Lampião is supposed to break into some of these houses. After breaking into a house, Lampião may **earn** or **lose** coins (some houses are guarded by *jagunços*, which are dangerous mercenaries crazy for money). More specifically, let C be how many coins Lampião possesses before breaking into the i-th house. Lampião will have $C + a_i$ coins after breaking into it, where a_i may actually be negative.

Lampião is initially standing in front of one of the houses. Also, he initially possesses 10^9 coins. The game is played in turns. The turns are numbered starting from one. In the i-th turn, one of the two actions below can be taken:

1. Move Lampião $3i$ houses to the right while breaking into **every** one of them, **except** for the final one. For instance, if this is the first turn, there are 4 houses, Lampião is standing by house 1 and this action is taken, Lampião will move to house 4 while breaking into houses 1, 2 and 3;
2. Just move Lampião $3i$ houses to the right.

The game instantly ends when Lampião goes beyond the n-th house.

Your task is to compute, for every possible starting position, the maximum profit Lampião can achieve if João acts optimally. The profit is defined as the difference between the final amount of coins and the initial amount of coins. Notice that the optimal profit is never negative, since João can choose not to take action (1).

Input

The first line contains an integer n ($1 \leq n \leq 50000$) – the number of houses in the mini-game.

The second line contains n space-separated integers. The i-th of them is a_i ($-300 \leq a_i \leq 300$).

Output

Print n lines. The i-th of them should contain a single integer – the maximum possible profit if Lampião starts from the i-th house.

Samples

standard input	standard output
5 1 2 3 4 -5	6 9 2 0 0

standard input	standard output
7 -3 -5 -7 -9 9 -2 -4	0 3 0 0 3 0 0

standard input	standard output
9 1 2 3 4 5 6 5 -7 2	21 20 18 15 16 6 0 0 2

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

Problem L

Code Name Hummingbird

Time limit: 2.5 seconds
Memory limit: 256 megabytes

The SIA (Soteropolis Intelligence Agency) has just engaged in a new operation code-named *Hummingbird*. It is responsible for the investigation of illegal biological activities in greater Soteropolis area. Tico is the one responsible for building a team for the operation. n agents will be designated to it, but first they must be given code names and then be arranged hierarchically.

More specifically, Tico will arrange those n agents in a line and number them from 1 to n, left to right. Right after that he will pick a code name for each one of the agents. The agent i will receive a **non-empty** code name with **at most** a_i characters. A code name is a string composed of:

- Zero or more digits from 0 to 9;
- **At least one** character from a set S of special characters (this set doesn't contain digits from 0 to 9). The characters of this set form a total order. Therefore, given two distinct special characters a, b, we can say that $a < b$ or $b < a$.

Tico says that the *strong character* of a code name is the special character which is greater than all the others. For example, if we take S as the set of English characters, the strong character of `4ak2k` is `k`.

After that, Tico will choose a direct supervisor for each of the agents in such a way that the following properties are met:

- Each agent should have as a supervisor another agent, except for one of them, which will be the leader of the operation;
- Every agent should directly supervise **at most** two other agents;
- An agent can't (directly or indirectly) supervise himself. Therefore, the supervising relation forms a rooted binary tree;
- For each agent i, let $s_1, s_2, \ldots, s_k$ be the **ordered** sequence of agents directly or indirectly supervised by him. If s is non-empty, then $max(s_k, i) - min(s_1, i)$ should be equal to k. In other words, every sub-tree of this tree must be formed by agents from a contiguous subsequence of $1, 2, \ldots, n$;
- The strong character of an agent's code name should be no smaller than the strong characters of the code names of the agents he supervise.

Tico defines a team as a pair of sequences (g, p) such all the properties above are met, where g_i is the code name of the i-th agent and p_i is the supervisor of the i-th agent (or 0 if he is the team leader).

Tico is interested in counting how many **distinct** teams he can get by picking code names and supervisors for the agents. More specifically, he wants to count that for many different sets S.

Two teams $(g, p), (t, s)$ are considered different iff there is an agent i such that $g_i \neq t_i$ **or** $p_i \neq s_i$.

Input

The first line contains two integers n, Q ($1 \leq n \leq 50$; $1 \leq Q \leq 50000$) – the number of agents and the number of sets S to be considered.

The second line contains n integers. The i-th of them is a_i ($1 \leq a_i \leq 5$) – the maximum length of the code name of the i-th agent.

Then Q lines follow. The i-th of them contains the size $|S_i|$ of a set of special characters ($1 \leq |S_i| \leq 10^9$).

Output

Print Q lines with a single integer in each one of them. The i-th of those should contain how many distinct teams can be formed considering S_i as the set of special characters.

Since those numbers can be huge, you should print the module $10^9 + 7$.

Samples

standard input	standard output
1 2 1 1 2	1 2

standard input	standard output
1 3 2 1 2 3	22 46 72

standard input	standard output
3 2 2 3 1 5 10	2540038 28724476

Note

In the first case, the only possible code names for $|S| = 2$ are `a` and `b`.

In the second case, for $|S| = 1$ we have the following strings in *regex* notation:

1. `a`
2. `a[0-9]`
3. `[0-9]a`
4. `aa`

Summing up to 22 possible code names.

Problem M

Sorting Machine

Time limit: 1.5 seconds
Memory limit: 256 megabytes

Betinho, the mayor of Soteropolis, striked (the public coffers) again! Now he bought a *Sorting Machine*.

A *Sorting Machine* is a dedicated processor with 1000 cores and a shared memory capable of storing an array of integers. In one operation, one can select up to 1000 **disjoint** intervals of this array and sort them concurrently. The cost of such operation is the maximum between the costs of sorting each of these intervals independently.

The cost of sorting a single interval is $L\lceil \log_2(K+1) \rceil$, where L is the size of the interval and K is the number of *adjacent inversions* it has. An adjacent inversion is a valid index i such that $a_i > a_{i+1}$.

Netinho wants to use his machine to sort the sequence $n, n-1, n-2, \ldots, 2, 1$. He wants to do at most 10 operations to sort this sequence, and he wants the total cost to be no more than $7n$. Find him a valid sequence of operations.

Input

The first line contains a single integer n ($4 \leq n \leq 10^5$).

Output

The first line should contain the number of operations O to be executed.

Then there should be O blocks describing each operation.

A operation block should contain in its first line an integer K ($1 \leq K \leq min(n, 1000)$) – the number of disjoint intervals used in this operation.

The next K lines of the block should contain two integers L, R ($1 \leq L \leq R \leq n$) – the boundaries of the interval to be sorted.

Samples

standard input	standard output
4	2 2 1 2 3 4 1 1 4

standard input	standard output
5	3 3 1 2 3 4 5 5 2 1 4 5 5 1 1 5