Problem A. Greedy Bipartite Matching

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Consider a bipartite weighted graph with $2n$ vertices: n in the left part and n in the right part. The vertices in each part are numbered from 1 to n. A matching is called *greedy* if it has the maximal number of edges of weight 1 among all matchings, the maximal number of edges of weight 2 among all matchings that maximize the number of edges of weight 1, etc.

Your task is to find the size (number of edges) of greedy matching in a dynamically growing graph.

Input

The first line of the input contains two non-negative integers n and q ($n \le 10^5$, $q \le 10^3$): the number of vertices in each part and the number of different weights of the edges.

Then, the input consists of q blocks. The i-th block starts with a non-negative integer m_i: the number of edges of weight i. Each of the next m_i lines contains two integers x and y ($1 \le x, y \le n$), which add an edge between vertex x of the left part and vertex y of the right part. It is guaranteed that $\sum_i m_i \le 2 \cdot 10^5$.

Note that there may be multiple edges between two vertices.

Output

You have to output q integers on a single line: answers for the problem for weights at most 1, weights at most 2, $\ldots$, weights at most q.

Example

standard input	*standard output*

3 4
2
1 1
1 2
2
1 1
2 2
2
1 3
3 2
1
3 3
	1 2 2 3

Problem B. Banshee

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are playing matchup Terran vs. Protoss in StarCraft II and the game came to elimination, which can be simplified to one-dimensional setting.

You have the army consisting of m bashees with upgraded hyperflight rotors, located at point 0 on a coordinate line. Your opponent has n buildings located on the positive half of the line. Each building is regarded as a segment.

Here are movement rules for banshees:

- A banshee with upgraded hyperflight rotors has a speed of 5.25 units per second. Acceleration is immediate.
- There is no unit collision for flying units in StarCraft, so any number of banshees can be located at the same point.

Next are the attack rules:

- A banshee can attack a target within a range of 6 units. A building can be attacked if any of its points is within the range.
- For simplicity, an attack goes as follows. First, the banshee has to wait without moving, for the whole cooldown time of 0.89 seconds. After that, it fires projectiles which immediately damage the target. This is the closest analogy of charging your weapons before the shot. Note that it is different from the actual StarCraft mechanics.
- In each attack, a banshee attacks one target and fires two projectiles at once. Each projectile deals 12 damage.

Finally, here are defense rules:

- Initially, each building has full hitpoints and full shields. A building is destroyed if its hitpoints drop to zero or below.
- If a building is damaged but not destroyed, and 10 seconds pass without taking damage, its shields begin to recharge. Such building recovers 2 shields per second until its shields are full or it is attacked again. The recovery is continuous: for example, 0.2 shields are recovered in 0.1 seconds.
- Assume a building has h hitpoints and s shields, and receives damage d. First, shields absorb all the damage they can: they are decreased by $d' = \min(s, d)$. Then, hitpoints are decreased by the remaining damage, $d - d'$.

You are given the initial position in this game. What is the minimal time required to eliminate all Protoss buildings?

Input

You will be given multiple test cases in this problem.

The first line contains a single integer t, the number of test cases. Then, you will be given t test cases in the format below.

The first line of each test case contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 10^9$): the number of remaining Protoss buildings and the number of your banshees. All your banshees are initially located at point 0 of the line.

Each of the following n lines contains four integers ℓ, r, h, s ($0 \le \ell < r \le 10^{12}$, $0 < h \le 10^6$, $0 \le s \le 10^6$): the left and right endpoints of the building, its hitpoints and its shields, respectively. You may assume that buildings don't overlap and are given in the order of increasing their ℓ coordinate.

The sum of n over all test cases is at most $2 \cdot 10^5$.

Output

For each test case, output the only number on a separate line: the minimal time required to eliminate all Protoss buildings. Your answer would be considered correct if its absolute difference with the jury's answer is at most 10^{-4}.

Example

standard input	*standard output*

2
2 1
1 2 1 100
100 500 736 0
3 2
0 1 12 0
1 2 6 6
2 3 3 10
	49.94476
	1.78000

Problem C. Courses

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	1024 mebibytes

Little Misha wants to change his IQ (initially he has 0 IQ). He found m types of courses on the internet. The i-th course type costs c_i bitcoins, changes his IQ by d_i (d_i can be negative, that is, his IQ can decrease after a course), and there are n_i different courses of i-th type. Authors of courses want to earn money, so $c_i \geq |d_i|$.

Misha wants to reach at least k IQ (of course, k can be negative). In order to achieve his goal, he will take a single course every day till some day. A course could be taken multiple times and each time it will affect Misha's IQ.

Now, he has n bitcoins. He is wondering: in how many ways can he spend exactly t bitcoins and reach at least k IQ in the end, for each $1 \leq t \leq n$? Two ways are considered different if they differ in the number of days to study or in a course taken at some day (different courses of the same type are considered different as well).

Input

The first line contains a single integer m ($0 < m < 100$): the number of types of courses.

Each of the next m lines contains three integers c_i, d_i, n_i ($0 < c_i < 10$, $|d_i| \leq c_i$, $0 \leq n_i \leq 10^4$).

And finally, the last line contains two integers n and k ($|k| \leq n \leq 3 \cdot 10^4$, $n > 0$).

Output

Output n integers, each on a separate line. The number on the i-th line should be the number of ways to spend exactly i bitcoins and obtain at least k IQ. Since these numbers can be large, output them modulo 998 244 353.

Examples

standard input	*standard output*

1
1 1 2
5 2
	0
	4
	8
	16
	32

2
1 -1 1
1 1 2
4 2
	0
	4
	8
	48

Problem D. Two Missing Numbers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 mebibytes

This is a run-twice problem: your solution will be executed twice on each test. See the rest of the statement and the input format section for more details.

You are a given a stream of integers such that each integer in the stream appears exactly twice, except for exactly two integers both of which appear exactly once. Your task is to construct a streaming algorithm that finds these two integers.

Input

Your solution will be invoked on each test twice.

On each invocation, the first line contains two integers q and n ($q \in \{1, 2\}$, $0 \le n \le 10^6$): the number of invocation and the size of the part of the stream. Additionally, on the second invocation, the first line also contains two integers x and y: the output of your program after the first invocation in the same exact order.

The second line contains n 64-bit unsigned integers separated by a space: the part of the stream itself.

In the first and the second parts combined, out of the integers that appear at all, each integer appears exactly twice, except for exactly two integers both of which appear exactly once.

Output

Your program has to print two 64-bit unsigned integers. The numbers after the first invocation will be given to your program on the second invocation in the same exact order. The numbers after the second invocation should be the answer (order is not important).

Examples

standard input	*standard output*

`1 5`
`5 1 4 4 5`
	`1 736`

`2 3 1 736`
`9 9 3`
	`1 3`

Problem E. Elimination Race

Input file:	*standard input*
Output file:	*standard output*
Time limit:	0.7 seconds
Memory limit:	1024 mebibytes

You are participating in an elimination race championship. The championship is held for n participants with assigned numbers from 1 to n and consists of $n-1$ races on different tracks. After each race, the participant who comes the last is eliminated. After all races, the only remaining participant is crowned as the champion.

As an experienced participant, you know how fast each car can go on each track, and therefore, you can predict the results of each race. However, the order of tracks is not yet determined. Your task is to find out for every participant whether there exists a permutation of the tracks that can grant him a victory.

Input

The first line contains the only integer n ($2 \le n \le 500$): the number of participants.

Each of the following $n-1$ lines describes a track and contains a permutation of numbers from 1 to n: the order in which the participants get to the finish on that track, from fastest to slowest.

Output

You should print n answers. The i-th answer should have the following format.

If winning the championship is possible for i-th participant, you should print "`Yes`" (without quotes, case insensitive) on the first line. On the second line, print a permutation of integers from 1 to $n-1$: the order of tracks which grants victory to the i-th participant. Tracks are numbered in the order they are given in the input.

Otherwise, you should output "`No`" on a single line.

If multiple answers exist, print any one of them.

Example

standard input	*standard output*

4
1 2 3 4
2 1 3 4
4 3 1 2
	Yes
	3 2 1
	No
	No
	No

Problem F. Forever Young

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Little Misha plays with infinite arrays which consist of nonnegative integers. Let us call such an array *good* if it is non-increasing.

In one step, Misha can increase or decrease one number in a good array by 1, if the array will remain good after this operation as well.

Initially, Misha had an array A. Misha made k steps and obtained an array B. In how many ways he could have obtained it?

Input

The first line contains a single integer n ($0 \le n \le 60$): the number of nonzero elements in A. The second line contains n integers separated by spaces: $60 \ge a_1 \ge a_2 \ge \cdots \ge a_n > 0$, the elements themselves. All other elements of A are zeroes.

The next two lines contain a description of B in the same format.

Additionally, it is guaranteed that $0 \le \sum a_i \le 60$ and $0 \le \sum b_i \le 60$.

The last line contains the only integer k ($0 \le k \le 10^6$).

Output

Print the desired number of ways modulo prime number 998 244 353.

Examples

standard input	*standard output*

3
3 2 1
3
3 2 1
2
	7

3
3 2 1
3
3 2 1
1111
	0

Note

In the first sample, the ways are:
$\{3,2,1\} \to \{4,2,1\} \to \{3,2,1\}$,
$\{3,2,1\} \to \{3,3,1\} \to \{3,2,1\}$,
$\{3,2,1\} \to \{3,2,2\} \to \{3,2,1\}$,
$\{3,2,1\} \to \{3,2,1,1\} \to \{3,2,1\}$,
$\{3,2,1\} \to \{2,2,1\} \to \{3,2,1\}$,
$\{3,2,1\} \to \{3,1,1\} \to \{3,2,1\}$,
$\{3,2,1\} \to \{3,2\} \to \{3,2,1\}$.

In the second sample, it is impossible to obtain the second array from the first in 1111 steps.

Problem G. Good and Lucky Matrices

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

This is a run-twice problem: your solution will be executed twice on each test. See the rest of the statement and the input format section for more details.

A square binary matrix is *good* if and only if it has an odd determinant.

A binary matrix of size $n \times n$ is *lucky* if and only if the greedy matching algorithm described below successfully finds a matching of size n. The algorithm reads the matrix line-by-line, from top to bottom. On each line, if there is a 1 in a column that was never chosen before, the algorithm greedily chooses the leftmost such column.

Your task is to compare numbers of good and lucky matrices of size $n \times n$. In order to do that, you have to produce a bijection between the smaller set and an equinumerous subset of the larger set.

Input

On each run, the first line contains an integer t: the number of test cases. The test cases follow.

The first line of each test case contains a string "`good`" or "`lucky`" denoting the type of matrix.

The next line contains a single integer n ($1 \le n \le 2000$): the size of the matrix.

Then follow n lines: i-th of them contains a binary string of length exactly n denoting the i-th row of the matrix.

The matrices given to your program on the second run will be the exactly the same as the matrices printed by your program on the first run, in the same order.

In is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

On each run, you have to print t answers. Each answer should be either a single integer -1 printed on a separate line or a matrix of the opposite type formatted similarly to the input: a line with size n followed by n lines describing the rows.

The mapping produced by your program should be a bijection. In other words, all the matrices printed on the second run should correspond to preimage of the corresponding matrix on the first run. Additionally, the number of -1 should be not greater than the difference in the sizes of the sets.

Formally, if on the first run your answer for matrix A of some type was matrix B, and on the second run your answer for matrix B of the opposite type was matrix C, then $A = C$ must hold. Additionally, for each n and each matrix type, let U be the set of all $n \times n$ matrices of this type, and V the set of $n \times n$ matrices of the other type. Then you can answer -1 for at most $\max(|U| - |V|, 0)$ different matrices from U.

Examples

standard input	*standard output*
3	2
lucky	11
2	10
11	2
11	11
good	01
2	2
11	01
01	10
lucky	
2	
01	
10	
3	2
good	11
2	11
11	2
10	11
lucky	01
2	2
11	01
01	10
good	
2	
01	
10	

Problem H. Shared Memory Switch

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Your task is to build an optimal algorithm for the shared memory switch.

A shared memory switch is one of the simplest nontrivial buffering architectures considered in the field of algorithms for networking. In this problem we consider a shared memory switch with multiple output queues and uniform (identical) packets. Incoming packets in this model are destined to one of the several output queues, which share a common buffer of finite size B. At the end of each second all nonempty queues transmit one packet each. When the buffer overflows your algorithm must decide which packet to drop. Your goal is to design an algorithm achieving maximal throughput (equivalently, dropping as few packets as possible).

You will be given n queries in the following format:

- -1: a single second passes.
- k $(1 \le k \le n)$: a packet arrives to the k-th queue.

You can assume that all packets remaining in the buffer after all queries will be sent as well or, equivalently, that there is an infinite number of -1 queries that follow the queries in the input.

Input

The first line contains contains two non-negative integers n and B $(n, B \le 2 \cdot 10^5)$: the number of queries and the amount of memory available to the switch correspondingly.

The second line contains n queries in the format described above.

Output

On the first line, output the only integer: the number of transmitted packets.

On the second line, output the numbers of queries (1-based) with arrivals of the packets that were transmitted. You can output these numbers in any order.

Examples

standard input	*standard output*
14 5 1 1 1 2 2 2 -1 1 -1 1 -1 1 -1 1	9 3 6 8 5 10 4 12 14 2
14 4 1 1 1 2 2 2 -1 1 -1 1 -1 1 -1 1	8 3 6 8 5 10 4 12 14

Problem I. Endless Road

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Suppose we have three chips on integer points on an infinite line (it is possible that two or more chips are at the same point). Every second one chip, taken equiprobably, moves to the next integer point (if the point was equal x, it becomes $x + 1$).

For each value of t from 1 to n, your task is to find the expected value of the maximal chip coordinate after t seconds.

Input

The first line of input contains three integers a, b, c ($0 \le a \le b \le c \le 10^6$): the initial coordinates of the chips.

The second line contains a single integer n ($1 \le n \le 2 \cdot 10^6$): the maximal time we are interested in.

Output

For each t from 1 to n, print a single line with a single number: the expected value of the maximal chip coordinate after t seconds, expressed as an integer modulo prime number 998 244 353. Formally, you can see that the expectation is a rational number $\frac{p}{q}$, where q is coprime with 998 244 353. You should output the number pq^{-1} modulo 998 244 353.

Examples

standard input	*standard output*
0 0 0 5	1 332748119 554580198 813384290 110916042
111 222 456 10	332748574 665496692 457 332748575 665496693 458 332748576 665496694 459 332748577

Problem J. Sets May Be Good

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Consider an undirected graph G with n vertices. A subset of its vertices is *good* if the total number of edges between them (edges such that both their ends are in this subset) is even. How many good sets are there? Since this number may be large, output it modulo prime number 998 244 353.

Input

The first line contains two integers n and m ($1 \le n \le 1000$, $0 \le m \le \frac{n(n-1)}{2}$): the number of vertices and edges in the graph, respectively.

Each of the following m lines contains two numbers u and v ($1 \le u, v \le n$): the vertices connected by an edge.

The graph is guaranteed to contain no loops or multiple edges.

Output

Output the number of good sets modulo 998 244 353.

Examples

standard input	*standard output*

5 5
1 2
2 3
3 4
4 5
1 5
	16

3 0	8
2 1 1 2	3

Note

In the second example, all sets are good. In the third example, the only non-good set is $\{1, 2\}$.

Problem K. Text Editor

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	150 mebibytes

This is a run-twice problem: your solution will be executed twice on each test. See the rest of the statement and the input format section for more details.

Your task is to implement a simple text editor that works with strings of characters with values in range 33–126 and implements an interface described below. The length of the string is guaranteed to be at most $2^{63}-1$ at all times. All parameters of the commands are 0-indexed and all intervals are semi-open (thus, whenever the command accepts ℓ and r, ℓ is included while r is excluded) and guaranteed to be nonempty. All positions and intervals are within bounds. The clipboard is initially empty.

- `insert` p str. This command inserts string str to the p-th position of the file. The total length of all str is guaranteed to be at most 10^6.
- `erase` ℓ r. This command erases the contents of the file from ℓ-th to r-th character.
- `print` ℓ r. After this command you should print the contents of the file from ℓ-th to r-th character. It is guaranteed that the total length of the output for all `print` commands does not exceed 10^6.
- `copy` ℓ r. This command copies the contents of the file from ℓ-th to r-th character to the clipboard.
- `cut` ℓ r. This command cuts the contents of the file from ℓ-th to r-th character to the clipboard.
- `paste` p. This commands inserts the contents of the clipboard to the p-th position of the file. It is guaranteed that empty clipboard is never pasted.
- `serialize`. After this command you should write the contents of the file (but not the clipboard) to the output, not exceeding 10^7 characters and using only characters of the range 33–126. The output string does not have to represent the contents of the file exactly, but your program should be able to recreate it in response to the command `deserialize`. You are also required to print **at least one** character. This command will be issued only once, as the last command on the first run.
- `deserialize` str. Here str is guaranteed to be the exact string that was printed in response to the `serialize` command. This command will be issued only once, as the first command on the second run.
- `undo`. After this command you should rollback the changes made by the last command changing the contents of the file (thus, `insert`, `erase`, `cut`, `paste` or `deserialize`), leaving the state of the clipboard intact. If there is no previous version, the command should do nothing.
- `redo`. Undo the last undo that is not yet undone. If there is no next version, the command should do nothing. Note that as in most text editors, the commands changing the state of the file also "erase" all next versions, but other commands (for example, `copy`) don't.

Input

Input formats for both runs are the same.

The first line contains the only integer n ($1 \le n \le 10^4$) representing the number of commands to the text editor.

The next n lines contain commands in the format described above.

Output

Print answers to all `print` and `serialize` commands on separate lines.

Examples

standard input	*standard output*

`17`
`insert 0 abcdef`
`print 0 6`
`erase 4 5`
`print 0 5`
`copy 0 3`
`paste 1`
`print 0 8`
`cut 2 4`
`print 0 6`
`undo`
`print 0 8`
`paste 6`
`print 0 10`
`redo`
`redo`
`print 0 10`
`serialize`
	`abcdef`
	`abcdf`
	`aabcbcdf`
	`aabcdf`
	`aabcbcdf`
	`aabcbcbcdf`
	`aabcbcbcdf`
	`serialize:aabcbcbcdf`

`2`
`deserialize serialize:aabcbcbcdf`
`print 0 10`
	`aabcbcbcdf`

Problem L. LCSLCSLCS

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Suppose we have two non-empty strings A and B ($|A|, |B| \leq 500$) of capital English letters, and two integers n and m such that $1 \leq n, m \leq 10^{15}$.

Let string A^n be a concatenation of n copies of string A. Let string B^m be a concatenation of m copies of string B. Your task is to find the longest common subsequence of A^n and B^m.

Input

On the first line, there are two integers n and m ($1 \leq n, m \leq 10^{15}$).

On the second line, there is a non-empty string A with length at most 500.

On the third line, there is a non-empty string B with length at most 500.

Both strings consist of capital English letters.

Output

Output one integer: the length of the longest common subsequence of A^n and B^m.

Examples

standard input	*standard output*

`10 10`
`AB`
`BA`
	`19`

`100000000 100000000`
`A`
`AB`
	`100000000`