A. It's Time To Duel

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

Something you may not know about Mouf is that he is a big fan of the Yu-Gi-Oh! card game. He loves to duel with anyone he meets. To gather all fans who love to play as well, he decided to organize a big Yu-Gi-Oh! tournament and invited n players.

Mouf arranged the n players in a line, numbered from 1 to n. They then held $n-1$ consecutive duels: for each i from 1 to $n-1$, player i faced player $i+1$, producing one winner and one loser per match. Afterward, each player reports a value $a_i (0 \le a_i \le 1)$:

- 0 indicating they won no duels;
- 1 indicating they won at least one duel.

Since some may lie about their results (e.g., reporting a 1 instead of a 0, or vice versa) to influence prize outcomes, Mouf will cancel the tournament if he can prove any report to be false.

Given the array a, determine whether at least one player must be lying.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains one integer n ($2 \le n \le 100$) — the number of players in the tournament.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$) — denoting the report of the i-th player.

Output

For each test case, print `"YES"` (without quotes) if there is at least one liar among the players, and `"NO"` (without quotes) otherwise.

You can output the answer in any case (upper or lower). For example, the strings `"yEs"`, `"yes"`, `"Yes"`, and `"YES"` will be recognized as positive responses.

Standard Input	Standard Output
6 3 0 1 0 2 0 0 2 1 1 4 0 1 1 1 4 1 0 0 1	NO YES YES NO YES NO

7 0 1 0 1 0 1 0	

Note

In the first test case, it is consistent if player 2 defeats both players 1 and 3, so nobody's report is necessarily false.

In the second test case, in the only match between players 1 and 2, one must win — but both claimed zero wins, so someone must be lying.

In the third test case, the tournament consists of exactly one duel between players 1 and 2 — but it's impossible for both to win, concluding that at least one report is false.

In the fourth test case, a possible scenario is that player 2 won against player 1, then 3 won against 2, and then 4 won against 3. All reports align, so there is no evidence that someone lied.

,

B. Slice to Survive

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

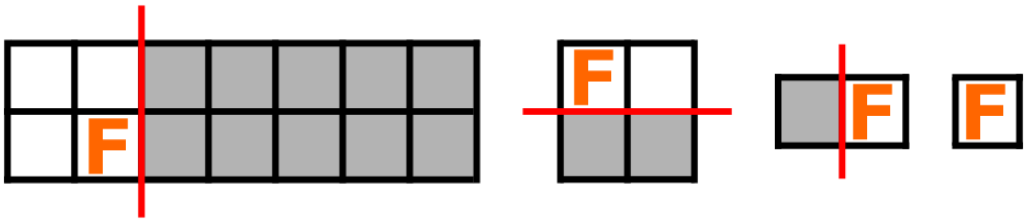
Duelists Mouf and Fouad enter the arena, which is an $n \times m$ grid!

Fouad's monster starts at cell (a, b), where rows are numbered 1 to n and columns 1 to m.

Mouf and Fouad will keep duelling until the grid consists of only one cell.

In each turn:

- Mouf first cuts the grid along a row or column line into two parts, discarding the part without Fouad's monster. Note that the grid must have at least two cells; otherwise, the game has already ended.
- After that, in the same turn, Fouad moves his monster to any cell (possibly the same one it was in) within the remaining grid.



Visualization of the phases of the fourth test case.

Mouf wants to minimize the number of turns, while Fouad wants to maximize them. How many turns will this epic duel last if both play optimally?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first and only line of each test case contains four integers n, m, a, and b ($2 \le n, m \le 10^9$, $1 \le a \le n$, $1 \le b \le m$) — denoting the number of rows, the number of columns, the starting row of the monster, and the starting column of the monster, respectively.

Output

For each test case, output a single integer — the number of turns this epic duel will last if both play optimally.

Standard Input	Standard Output
8 2 2 1 1 3 3 2 2 2 7 1 4 2 7 2 2 8 9 4 6 9 9 5 5 2 20 2 11 22 99 20 70	2 4 4 3 6 8 6 10

Note

In the first test case, one possible duel sequence is as follows:

- Turn 1: Mouf cuts the grid horizontally along the line between the rows 1 and 2, removing the bottom half and leaving a 1×2 grid.
- Turn 1: Fouad's monster is at the cell $(1, 1)$.
- Turn 2: Mouf cuts the 1×2 grid again, removes one column, and isolates the cell $(1, 1)$.

The duel is completed in 2 turns.

In the fourth case, one possible duel sequence is as follows:

- Turn 1: Mouf cuts the grid vertically along the line between the columns 2 and 3, splitting it into a 2×2 and a 2×5 field, then removes the 2×5 part.
- Turn 1: Fouad moves the monster to the cell $(1, 1)$.
- From this point on, the duel plays out just like the first test case—two more turns trim down the grid from 2×2 to a single 1×1 cell.

In total, the duel is completed in 3 turns.

You can refer to the pictures mentioned in the problem statement for illustrations of the fourth test case.

,

C1. Hacking Numbers (Easy Version)

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

This is the easy version of the problem. In this version, you can send at most 7 commands. You can make hacks only if all versions of the problem are solved.

This is an interactive problem.

Welcome, Duelists! In this interactive challenge, there is an unknown integer x ($1 \le x \le 10^9$). You must make it equal to a given integer in the input n. By harnessing the power of "Mathmech" monsters, you can send a command to do one of the following:

Command	Constraint	Result	Case	Update	Jury's response
"add y"	$-10^{18} \le y \le 10^{18}$	$\text{res} = x + y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"mul y"	$1 \le y \le 10^{18}$	$\text{res} = x \cdot y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"div y"	$1 \le y \le 10^{18}$	$\text{res} = x/y$	if y divides x	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"digit"	—	$\text{res} = S(x)^*$	—	$x \leftarrow \text{res}$	"1"

You have to make x equal to n using **at most 7** commands.

---

* $S(n)$ is a function that returns the sum of all the individual digits of a non-negative integer n. For example, $S(123) = 1 + 2 + 3 = 6$

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

The first and only line of each test case contains one integer n ($1 \le n \le 10^9$).

Interaction

The interaction for each test case begins by reading the integer n.

To send a command, output a line in the following format:

- "add y" Add some integer y ($-10^{18} \le y \le 10^{18}$) to x.
  The jury will output "1" if $x + y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x + y$.
- "mul y" Multiply x by a positive integer y ($1 \le y \le 10^{18}$).
  The jury will output "1" if $x \cdot y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x \cdot y$.
- "div y" Divide x by a positive integer y ($1 \le y \le 10^{18}$).

The jury will output `"1"` if y is a divisor of x (**successful**), and `"0"` otherwise. If successful, update $x \leftarrow \frac{x}{y}$.

- `"digit"` Make x equal to the sum of its digits.
  The jury will always output `"1"` and update $x \leftarrow S(x)$.

Note that commands are **case sensitive**.

When you have determined that x is equal to n, output a line in the following format:

- `"!"` — where the jury will output a `"1"` if n is equal to x, and `"-1"` otherwise.

Note that answering **does not count** toward your limit of 7 commands.

If your program makes more than 7 commands for one test case, or makes an invalid command, then the response to the command will be `"-1"`. After receiving such a response, your program should immediately terminate to receive the verdict **Wrong Answer**. Otherwise, it may receive any other verdict.

After printing a command, do not forget to output the end of the line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- `std::io::stdout().flush()` in Rust;
- see the documentation for other languages.

The interactor is **non-adaptive**. The unknown integer x **does not change** during the interaction.

Hacks

To hack, use the following format.

The first line should contain a single integer t ($1 \le t \le 5000$) — the number of test cases.

The first line of each test case should contain two positive integers n and x ($1 \le n, x \le 10^9$) — denoting the unknown integer and the target value to which it should be made equal, respectively.

Standard Input	Standard Output
2 100 0 1 1 1 5 1	add -10 add 1 mul 10 ! digit div 2 !

1 1	

Note

Solution	Jury	Explanation
	2	There are 2 test cases.
	100	In the first test case, the unknown integer $x = 9$ and we have to make it equal to $n = 100$.
add -10	0	The answer to "add -10" is "0". This means that the addition command was not successful as $x + y = 9 + (-10) \leq 0$, and x remains 9 after the command
add 1	1	The answer to "add 1" is "1". This means that the addition command was successful as $x + y = 9 + 1 = 10$, and x changes to 10 after the command.
mul 10	1	The answer to "mul 10" is "1". This means that the multiplication command was successful as $x \cdot y = 10 \cdot 10 = 100$, and x changes to 100 after the command.
!	1	The answer to "!" is "1". This means you have determined that x equals n.
	5	In the second test case, the unknown integer $x = 1234$ and we have to make it equal to $n = 5$.
digit	1	The answer to "digit" is "1". This means that x turned into the sum of its digits $1 + 2 + 3 + 4 = 10$, and x changes to 10 after the command.
div 2	1	The answer to "div 2" is "1". This means that the division command was successful as $y = 2$ is a divisor of $x = 10$, and x changes to $\frac{x}{y} = \frac{10}{2} = 5$ after the command.
!	1	The answer to "!" is "1". This means you have determined that x equals n.

Note that the empty lines in the example input and output are for the sake of clarity, and do not occur in the real interaction.

,

C2. Hacking Numbers (Medium Version)

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

This is the medium version of the problem. In this version, you can send at most 4 commands. You can make hacks only if all versions of the problem are solved.

This is an interactive problem.

Welcome, Duelists! In this interactive challenge, there is an unknown integer x ($1 \le x \le 10^9$). You must make it equal to a given integer in the input n. By harnessing the power of "Mathmech" monsters, you can send a command to do one of the following:

Command	Constraint	Result	Case	Update	Jury's response
"add y"	$-10^{18} \le y \le 10^{18}$	$\text{res} = x + y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"mul y"	$1 \le y \le 10^{18}$	$\text{res} = x \cdot y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"div y"	$1 \le y \le 10^{18}$	$\text{res} = x/y$	if y divides x	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"digit"	—	$\text{res} = S(x)^*$	—	$x \leftarrow \text{res}$	"1"

You have to make x equal to n using **at most 4** commands.

---

* $S(n)$ is a function that returns the sum of all the individual digits of a non-negative integer n. For example, $S(123) = 1 + 2 + 3 = 6$

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

The first and only line of each test case contains one integer n ($1 \le n \le 10^9$).

Interaction

The interaction for each test case begins by reading the integer n.

To send a command, output a line in the following format:

- "add y" Add some integer y ($-10^{18} \le y \le 10^{18}$) to x.
  The jury will output "1" if $x + y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x + y$.
- "mul y" Multiply x by a positive integer y ($1 \le y \le 10^{18}$).
  The jury will output "1" if $x \cdot y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x \cdot y$.
- "div y" Divide x by a positive integer y ($1 \le y \le 10^{18}$).

The jury will output `"1"` if y is a divisor of x (**successful**), and `"0"` otherwise. If successful, update $x \leftarrow \frac{x}{y}$.

- `"digit"` Make x equal to the sum of its digits.
  The jury will always output `"1"` and update $x \leftarrow S(x)$.

Note that commands are **case sensitive**.

When you have determined that x is equal to n, output a line in the following format:

- `"!"` — where the jury will output a `"1"` if n is equal to x, and `"-1"` otherwise.

Note that answering **does not count** toward your limit of 4 commands.

If your program makes more than 4 commands for one test case, or makes an invalid command, then the response to the command will be `"-1"`. After receiving such a response, your program should immediately terminate to receive the verdict **Wrong Answer**. Otherwise, it may receive any other verdict.

After printing a command, do not forget to output the end of the line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- `std::io::stdout().flush()` in Rust;
- see the documentation for other languages.

The interactor is **non-adaptive**. The unknown integer x **does not change** during the interaction.

Hacks

To hack, use the following format.

The first line should contain a single integer t ($1 \leq t \leq 5000$) — the number of test cases.

The first line of each test case should contain two positive integers n and x ($1 \leq n, x \leq 10^9$) — denoting the unknown integer and the target value to which it should be made equal, respectively.

Standard Input	Standard Output
2 100 0 1 1 1 5 1 	add -10 add 1 mul 10 ! digit div 2 !

1 1	

Note

Solution	Jury	Explanation
	2	There are 2 test cases.
	100	In the first test case, the unknown integer $x = 9$ and we have to make it equal to $n = 100$.
add -10	0	The answer to "add -10" is "0". This means that the addition command was not successful as $x + y = 9 + (-10) \leq 0$, and x remains 9 after the command
add 1	1	The answer to "add 1" is "1". This means that the addition command was successful as $x + y = 9 + 1 = 10$, and x changes to 10 after the command.
mul 10	1	The answer to "mul 10" is "1". This means that the multiplication command was successful as $x \cdot y = 10 \cdot 10 = 100$, and x changes to 100 after the command.
!	1	The answer to "!" is "1". This means you have determined that x equals n.
	5	In the second test case, the unknown integer $x = 1234$ and we have to make it equal to $n = 5$.
digit	1	The answer to "digit" is "1". This means that x turned into the sum of its digits $1 + 2 + 3 + 4 = 10$, and x changes to 10 after the command.
div 2	1	The answer to "div 2" is "1". This means that the division command was successful as $y = 2$ is a divisor of $x = 10$, and x changes to $\frac{x}{y} = \frac{10}{2} = 5$ after the command.
!	1	The answer to "!" is "1". This means you have determined that x equals n.

Note that the empty lines in the example input and output are for the sake of clarity, and do not occur in the real interaction.

,

C3. Hacking Numbers (Hard Version)

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

This is the hard version of the problem. In this version, the limit of commands you can send is described in the statement. You can make hacks only if all versions of the problem are solved.

This is an interactive problem.

Welcome, Duelists! In this interactive challenge, there is an unknown integer x ($1 \le x \le 10^9$). You must make it equal to a given integer in the input n. By harnessing the power of "Mathmech" monsters, you can send a command to do one of the following:

Command	Constraint	Result	Case	Update	Jury's response
"add y"	$-10^{18} \le y \le 10^{18}$	$\text{res} = x + y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"mul y"	$1 \le y \le 10^{18}$	$\text{res} = x \cdot y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"div y"	$1 \le y \le 10^{18}$	$\text{res} = x/y$	if y divides x	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"digit"	—	$\text{res} = S(x)^*$	—	$x \leftarrow \text{res}$	"1"

Let $f(n)$ be the minimum integer such that there is a sequence of $f(n)$ commands that transforms x into n for all $x (1 \le x \le 10^9)$. You do not know the value of x in advance. Find $f(n)$ such that, no matter what x is, you can always transform it into n using at most $f(n)$ commands.

Your task is to change x into n using **at most** $f(n)$ commands.

---

* $S(n)$ is a function that returns the sum of all the individual digits of a non-negative integer n. For example, $S(123) = 1 + 2 + 3 = 6$

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

The first and only line of each test case contains one integer n ($1 \le n \le 10^9$).

Interaction

The interaction for each test case begins by reading the integer n.

To send a command, output a line in the following format:

- "add y" Add some integer y ($-10^{18} \le y \le 10^{18}$) to x.
  The jury will output "1" if $x + y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x + y$.

- `"mul y"` Multiply x by a positive integer y ($1 \le y \le 10^{18}$).
  The jury will output `"1"` if $x \cdot y$ is within $[1, 10^{18}]$ (**successful**), and `"0"` otherwise. If successful, update $x \leftarrow x \cdot y$.
- `"div y"` Divide x by a positive integer y ($1 \le y \le 10^{18}$).
  The jury will output `"1"` if y is a divisor of x (**successful**), and `"0"` otherwise. If successful, update $x \leftarrow \frac{x}{y}$.
- `"digit"` Make x equal to the sum of its digits.
  The jury will always output `"1"` and update $x \leftarrow S(x)$.

Note that commands are **case sensitive**.

When you have determined that x is equal to n, output a line in the following format:

- `"!"` — where the jury will output a `"1"` if n is equal to x, and `"-1"` otherwise.

Note that answering **does not count** toward your limit of commands.

If your program makes more than $f(n)$ commands ($f(n)$ is described above) for one test case, or makes an invalid command, then the response to the command will be `"-1"`. After receiving such a response, your program should immediately terminate to receive the verdict **Wrong Answer**. Otherwise, it may receive any other verdict.

After printing a command, do not forget to output the end of the line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- `std::io::stdout().flush()` in Rust;
- see the documentation for other languages.

The interactor is **non-adaptive**. The unknown integer x **does not change** during the interaction.

Hacks

To hack, use the following format.

The first line should contain a single integer t ($1 \le t \le 5000$) — the number of test cases.

The first line of each test case should contain two positive integers n and x ($1 \le n, x \le 10^9$) — denoting the unknown integer and the target value to which it should be made equal, respectively.

Standard Input	Standard Output
2 100 0 1 1 	add -10 add 1 mul 10 ! digit

1 5 1 1 1	 div 2 !

Note

Solution	Jury	Explanation
	2	There are 2 test cases.
	100	In the first test case, the unknown integer $x = 9$ and we have to make it equal to $n = 100$.
add -10	0	The answer to "add -10" is "0". This means that the addition command was not successful as $x + y = 9 + (-10) \leq 0$, and x remains 9 after the command
add 1	1	The answer to "add 1" is "1". This means that the addition command was successful as $x + y = 9 + 1 = 10$, and x changes to 10 after the command.
mul 10	1	The answer to "mul 10" is "1". This means that the multiplication command was successful as $x \cdot y = 10 \cdot 10 = 100$, and x changes to 100 after the command.
!	1	The answer to "!" is "1". This means you have determined that x equals n.
	5	In the second test case, the unknown integer $x = 1234$ and we have to make it equal to $n = 5$.
digit	1	The answer to "digit" is "1". This means that x turned into the sum of its digits $1 + 2 + 3 + 4 = 10$, and x changes to 10 after the command.
div 2	1	The answer to "div 2" is "1". This means that the division command was successful as $y = 2$ is a divisor of $x = 10$, and x changes to $\frac{x}{y} = \frac{10}{2} = 5$ after the command.
!	1	The answer to "!" is "1". This means you have determined that x equals n.

Note that the empty lines in the example input and output are for the sake of clarity, and do not occur in the real interaction.

,

D. D/D/D

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

Of course, a problem with the letter D is sponsored by Declan Akaba.

You are given a simple, connected, undirected graph with n vertices and m edges. The graph contains no self-loops or multiple edges. You are also given a multiset A consisting of ℓ elements:

$$A = \{A_1, A_2, \ldots, A_\ell\}$$

Starting from vertex 1, you may perform the following move **any number** of times, as long as the multiset A is not empty:

- Select an element $k \in A$ and remove it from the multiset . You must remove exactly one occurrence of k from A.
- Traverse any walk* of exactly k edges to reach some vertex (possibly the same one you started from).

For each i ($1 \le i \le n$), determine whether there exists a sequence of such moves that starts at vertex 1 and ends at vertex i, using the original multiset A.

Note that the check for each vertex i is independent — you restart from vertex 1 and use the original multiset A for each case.

* A walk of length k is a sequence of vertices $v_0, v_1, \ldots, v_{k-1}, v_k$ such that each consecutive pair of vertices (v_i, v_{i+1}) is connected by an edge in the graph. The sequence may include repeated vertices.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, m, and ℓ ($2 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 4 \cdot 10^5$, $1 \le \ell \le 2 \cdot 10^5$) — the number of vertices, the number of edges, and the size of the multiset, respectively.

The second line of each test case contains ℓ integers $A_1, A_2, \ldots, A_\ell$ ($1 \le A_i \le 10^4$) — the elements of the multiset.

Each of the following m lines contains two integers u and v ($1 \le u < v \le n$) — the endpoints of an edge in the graph.

It is guaranteed that the edges form a simple, connected graph without self-loops or multiple edges.

It is guaranteed that the sum of n, the sum of m, and the sum of ℓ over all test cases does not exceed $2 \cdot 10^5$, $4 \cdot 10^5$, and $2 \cdot 10^5$, respectively.

Output

For each test case, output a binary string of length n, where the i-th character is `1` if there exists a sequence of moves ending at vertex i, and `0` otherwise.

Standard Input	Standard Output

3 6 5 2 2 3 1 2 2 3 3 4 4 5 5 6 5 5 1 5 1 2 2 3 3 4 4 5 3 5 5 4 3 100 200 300 1 2 1 3 1 4 2 5	111101 11111 10001

Note

In the first test case:

- Vertex 1 is reachable without making any moves.
- Vertex 2 is reachable by selecting element $3 \in A$; one possible walk is $[1 \to 2 \to 1 \to 2]$.
- Vertex 3 can be reached by selecting element $2 \in A$ and taking the walk $[1 \to 2 \to 3]$.
- Vertex 4 is reachable by selecting element $3 \in A$ and following the walk $[1 \to 2 \to 3 \to 4]$.
- Vertex 5 is not reachable by any valid sequence of moves.
- Vertex 6 is reachable by first selecting element $2 \in A$ and taking the walk $[1 \to 2 \to 3]$, followed by selecting element $3 \in A$ and taking the walk $[3 \to 4 \to 5 \to 6]$.

,

E. Binary String Wowee

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

Mouf is bored with themes, so he decided not to use any themes for this problem.

You are given a binary* string s of length n. You are to perform the following operation exactly k times:

- select an index i ($1 \leq i \leq n$) such that $s_i = 0$;
- then flip† each s_j for all indices j ($1 \leq j \leq i$).

You need to count the number of possible ways to perform all k operations.

Since the answer could be ginormous, print it modulo $998\,244\,353$.

Two sequences of operations are considered different if they differ in the index selected at any step.

---

*A binary string is a string that consists only of the characters 0 and 1.

†Flipping a binary character is changing it from 0 to 1 or vice versa.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 100$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \leq k \leq n \leq 500$) — the length of the binary string s and the number of times the operation must be performed, respectively.

The second line of each test case contains a binary string s of length n consisting of only characters 0 and 1.

It is guaranteed that the sum of n does not exceed 500 over all test cases.

Output

For each test case, output a single integer — the number of ways you can perform exactly k operations, modulo $998\,244\,353$.

Standard Input	Standard Output
5 3 1 010 3 2 000 5 4 01001 8 8 11001100 20 20 10010110101101010110	2 3 10 27286 915530405

Note

In the first test case, here are all the possible sequences of operations:

- $010 \xrightarrow{i=1} 110$
- $010 \xrightarrow{i=3} 101$

In the second test case, here are all the possible sequences of operations:

- $000 \xrightarrow{i=1} 100 \xrightarrow{i=2} 010$
- $000 \xrightarrow{i=1} 100 \xrightarrow{i=3} 011$
- $000 \xrightarrow{i=2} 110 \xrightarrow{i=3} 001$

,

F. Penguin Steps

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

Mouf, the clever master of Darkness, and Fouad, the brave champion of Light, have entered the Grid Realm once more. This time, they have found the exit, but it is guarded by fierce monsters! They must fight with their bare hands instead of summoning monsters!

Mouf and Fouad are standing on an $n \times n$ grid. Each cell (i, j) has a value $a_{i,j}$ and a color. The color of a cell is white if $c_{i,j} = 0$ and black if $c_{i,j} = 1$.

Mouf starts at the top-left corner $(1, 1)$, and Fouad starts at the bottom-left corner $(n, 1)$. Both are trying to reach the exit cell at (r, n).

A path is defined as a sequence of adjacent cells (sharing a horizontal or vertical edge). The cost of a path is the maximum value of $a_{i,j}$ among all cells included in the path (including the first and last cells).

Let:

- dis_M denote the minimum possible cost of a valid path from Mouf's starting position $(1, 1)$ to the exit (r, n);
- dis_F denote the minimum possible cost of a valid path from Fouad's starting position $(n, 1)$ to the exit (r, n).

Before moving, Mouf can perform up to k operations. In each operation, he may select any black cell and increment its value by 1 (possibly choosing the same cell multiple times).

Mouf wants to maximize dis_F while ensuring that his own cost dis_M remains **unchanged** (as if he performed no operations). If Mouf acts optimally, what are the values of dis_M and dis_F?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains three integers n, r, and k ($2 \le n \le 300$, $1 \le r \le n$, $0 \le k \le 10^6$) — the length of the grid, the row number of the exit cell, and the number of allowed operations.

The i-th of the next n lines contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($1 \le a_{ij} \le 10^6$) — the values of the cells in the i-th row.

The i-th of the next n lines contains a binary string c_i of length n — denoting the color of the cells in the i-th row (cell (i, j) is white if $c_{i,j} =$ `0` and black if $c_{i,j} =$ `1`).

It is guaranteed that the sum of n^2 over all test cases does not exceed $9 \cdot 10^4$.

Output

For each test case, output two integers — dis_M and dis_F if Mouf performs the operations optimally.

Standard Input	Standard Output

```
4
2 1 30
2 2
1 1
11
01
3 3 5
9 2 2
2 3 2
2 2 2
111
111
010
7 3 12
3 3 3 3 5 1 1
9 4 8 3 3 5 5
9 4 8 7 3 3 3
4 4 4 4 9 4 9
4 4 4 4 9 4 9
1 4 4 4 4 4 9
1 1 4 4 9 9 9
1111111
1011111
1011111
1111111
1111101
1110001
0111111
5 3 1419
1219 678 1672 1858 1210
535 732 1316 345 296
1106 3060 507 216 1943
194 2124 47 87 4818
1007 329 1425 284 660
00010
10111
00101
10001
10100
```

```
2 2
9 5
3 8
1943 2426
```

```
1
8 2 2216
429 589 675 2022 259 452 733 967
1097 2880 256 1894 259 1052 345 692
911 831 513 1243 200 14 854 217
611 882 681 279 54 719 1469 1885
504 2524 1332 17 3113 34 1281 717
498 1896 1800 2231 731 364 69 1247
1397 399 68 448 1337 1076 166 3786
```

```
733 1671
```

16 857 91 475 106 102 1517 1949 01010100 00101100 00001000 10100110 00001000 00100000 01100011 00001000	

Note

In the first test case:

- Although Mouf can perform up to 30 operations, he can not increase dis_F beyond 2; he is restricted to applying operations only on $(2, 2)$, because performing operations on $(1, 1)$ or $(1, 2)$ would change dis_M.
- Mouf may apply all 30 operations on cell $(2, 2)$; however, Fouad can still follow the path $(2, 1) \to (1, 1) \to (1, 2)$ with a cost of 2.

In the second test case, Mouf can apply two operations on $(2, 2)$ and three operations on $(3, 2)$. It can be shown that Mouf can not increase dis_F beyond 5.