The 4th Universal Cup

Extra Stage 5: Shenzhen

April 12-19, 2026

This problem set should contain 13 problems on 23 numbered pages.

Based on

International Collegiate Programming Contest (ICPC)

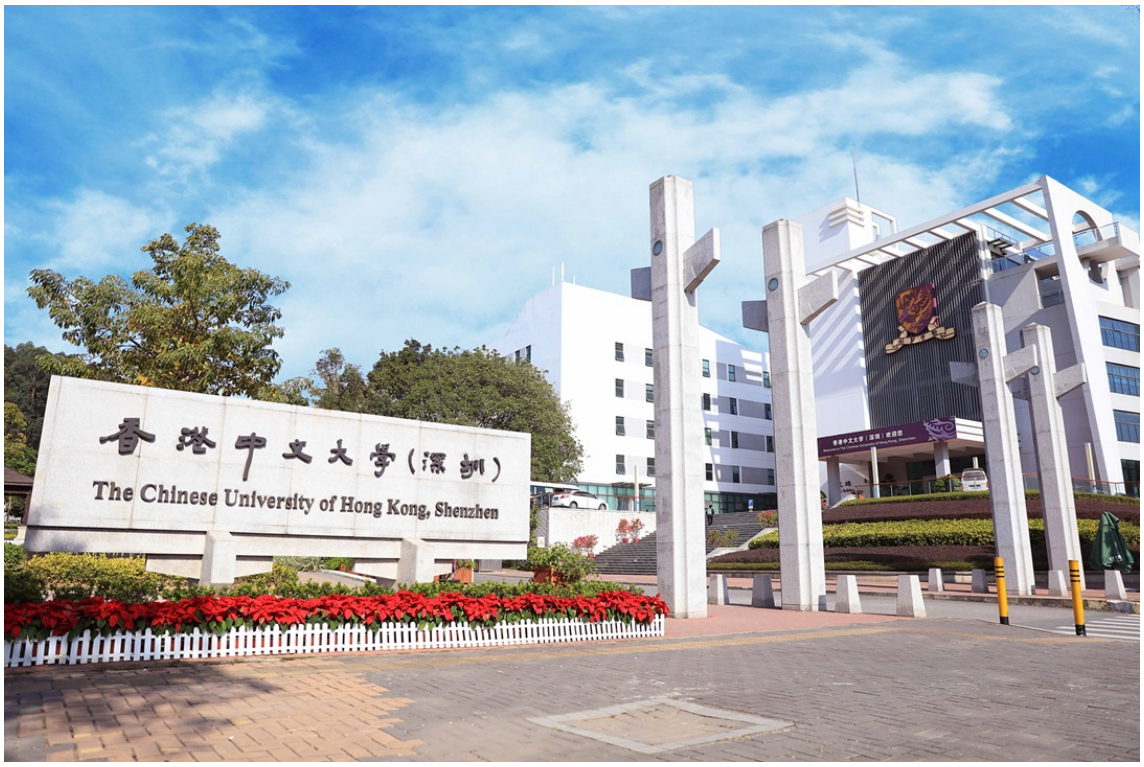
Hosted by

Prepared by

Problem A. Greetings from Prof. Chen

Time limit:	1 second
Memory limit:	1024 megabytes

Welcome to CUHK-Shenzhen! Prof. Chen loves to send greetings using strings consisting only of lowercase English letters.



Given an integer n, construct a string consisting only of lowercase English letters such that the sum of the ASCII codes of its characters is equal to n.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 5 \times 10^3$), indicating the number of test cases. For each test case:

The only line contains an integer n ($1 \le n \le 10^7$).

It is guaranteed that the sum of n over all test cases does not exceed 10^7.

Output

For each test case:

- If it is possible to construct a string consisting only of lowercase English letters whose ASCII sum is equal to n, first output `Yes` in one line. Then output a second line containing such a string.
- Otherwise, if it is impossible to do so, just output `No` in one line.

Example

standard input	standard output
3 2257 2269 96	Yes greetingsfromprofchen Yes welcometocuhkshenzhen No

Note

Below is the ASCII code table for lowercase English letters:

Letter	ASCII	Letter	ASCII	Letter	ASCII
a	97	j	106	s	115
b	98	k	107	t	116
c	99	l	108	u	117
d	100	m	109	v	118
e	101	n	110	w	119
f	102	o	111	x	120
g	103	p	112	y	121
h	104	q	113	z	122
i	105	r	114		

Problem B. All-Star Showdown

Time limit: 2 seconds
Memory limit: 1024 megabytes

In a bold departure from tradition, the Grand Contest this year will feature a special All-Star Showdown: a show match pitting two elite squads, Team Red and Team Blue, against each other in a head-to-head programming duel. Unlike the standard contest format, this match allows for flexible team sizes, as long as every contestant is assigned to exactly one team.

There are n all-star contestants selected from the hall of fame. Each contestant i is evaluated across three core dimensions: algorithmic insight, implementation speed, and debugging resilience. These attributes are quantified as positive integers (x_i, y_i, z_i), forming a 3D capability vector that captures their competitive profile. Additionally, each contestant i is assigned a positive integer synergy value w_i, reflecting their ability to amplify team performance. The overall team value of a squad is defined as the product of the synergy values of its members.

To maintain competitive integrity and prevent dominance by clusters of stylistically similar coders, the organizers enforce a strict diversity rule: within any single team, the Euclidean distance between the capability vectors of any two members must be at least d. Formally, for any two different contestants i and j in the same team, it must hold that $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \geq d$. This ensures that each team comprises a well-rounded mix of problem-solving approaches.

The organizing committee wishes to analyze the combinatorial landscape of all valid assignments. Specifically, for each integer $k = 1, 2, \ldots, n-1$, they request the sum of the Team Red values over all valid partitions in which Team Red contains exactly k contestants.

Since the answers may be large, output them modulo 998 244 353.

Input

There is only one test case in each test file.

The first line contains two integers n and d ($2 \leq n \leq 10^5$, $1 \leq d \leq 10^8$).

For the following n lines, the i-th line contains four integers x_i, y_i, z_i, and w_i ($1 \leq x_i, y_i, z_i, w_i \leq 10^8$), indicating the capability vector and synergy value of contestant i.

Output

Output $(n-1)$ lines, where the i-th line contains an integer, indicating the sum of the Team Red values over all valid partitions in which Team Red contains exactly i contestants modulo 998 244 353.

Examples

standard input	standard output
3 1 1 1 1 1 1 2 1 10 1 2 3 100	111 1110
3 2 1 1 1 1 1 2 1 10 1 2 3 100	11 1100
3 2 2 1 1 1 1 2 1 10 1 1 2 100	0 0

Problem C. One Item Away

Time limit:	1.5 seconds
Memory limit:	1024 megabytes

This is an interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

There are n people and m items. Each item must be assigned to exactly one person.

Different people may value items differently. For any collection of items, a person can evaluate their satisfaction with it. You do not know exactly how these evaluations are computed, but you are allowed to query their values. You may ask at most $n \times m$ queries.

Formally, for each person i, there exists a function $u_i(S)$ that assigns a non-negative integer value to any subset S of items. These functions satisfy:

- $u_i(\varnothing) = 0$;
- If $S \subseteq T$, then $u_i(S) \le u_i(T)$.

In other words, giving someone more items never makes them less satisfied.

However, people can be sensitive. If person i observes that another person j has a strictly better collection of items, they may start to complain.

Fortunately, people are also somewhat forgiving: if removing just one item from j's collection is enough to eliminate this advantage, then no complaint is made.

Formally, let s_i denote the set of items assigned to person i. Your goal is to assign every item to exactly one person so that, for every pair of people i and j, if person i considers j's collection strictly better than their own, i.e., $u_i(s_j) > u_i(s_i)$, then there exists an item $x \in s_j$ such that $u_i(s_j \setminus \{x\}) \le u_i(s_i)$.

Input

There is only one test case in each test file.

The first line contains two integers n and m ($1 \le n \le 100$, $1 \le m \le 500$), indicating the number of people and the number of items.

Interaction Protocol

You can evaluate values using queries.

To ask for the value that person i assigns to a subset of items $\{x_1, x_2, \ldots, x_k\}$, output one line:

$$\texttt{?}\ i\ k\ x_1\ x_2\ \ldots\ x_k$$

where $1 \le i \le n$, $0 \le k \le m$, and $x_1, x_2, \ldots, x_k$ are pairwise distinct integers satisfying $1 \le x_t \le m$.

After flushing your output, your program should read a single integer v ($0 \le v \le 10^9$), which is equal to $u_i(\{x_1, x_2, \ldots, x_k\})$.

You may ask at most $n \times m$ queries. Also note that the interactor is adaptive. In particular, the answers to your queries may depend on your previous queries, but they are always consistent with some valid set of functions $u_i(\cdot)$ satisfying all constraints in the statement.

When you are ready to assign items, output one line:

! a_1 a_2 ... a_m

where a_j is the person to whom item j is assigned, and $1 \le a_j \le n$. Assigning items does not count as a query.

After flushing your output, your program should terminate immediately.

Example

standard input	standard output
2 3 10 20 10 5	 ? 1 1 1 ? 1 2 2 3 ? 2 1 1 ? 2 2 2 3 ! 2 1 1

Problem D. City Management

Time limit:	3 seconds
Memory limit:	1024 megabytes

You are managing a city. Several factories will be established in your city, and several workers will come to your city looking for jobs. You need to arrange some affairs reasonably to maximize the total profit.

Each factory has a production capacity x_i, which means that in one round of production:

- If there is a worker working in it, it yields a profit of x_i.
- If there is no worker working in it, it yields no profit.
- At most one worker can work in each factory.

You have foreseen that in the next n days, each day one of the following two events will occur:

- A worker comes looking for a job. After this worker arrives, you can reassign the positions of all workers (including the workers currently in a factory). Specifically, for each worker, you can assign him/her to work in any factory or leave him/her idle.
- A factory with production capacity x_i is established. After this factory is established, if there are idle workers, you can choose whether to assign an idle worker to work in this factory. Of course, you can also leave the factory vacant. In particular, if there are no idle workers, this factory can only be left vacant.

After the event of the day is executed, all factories will conduct one round of production and yield profit according to the rules described above.

You need to find the maximum total profit.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^6$), indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 2 \times 10^5$), indicating the number of days.

For the following n lines, the i-th line first contains a character op_i ($op_i \in \{\texttt{W}, \texttt{F}\}$), indicating the type of event on the i-th day. If $op_i = \texttt{W}$, a worker comes looking for a job on that day; otherwise, if $op_i = \texttt{F}$, another integer x_i follows ($1 \le x_i \le 10^7$), indicating that a factory with production capacity x_i is established on that day.

It is guaranteed that the sum of n of all test cases does not exceed 10^6.

Output

For each test case, output one line containing one integer, indicating the maximum total profit.

Example

standard input	standard output
2 8 F 1 F 2 W F 100 W W F 10000 W 9 W F 51 F 1 F 1 F 1 F 1 F 100 F 100 W	20509 557

Note

We explain the first sample test case below:

Day	Action	Profit of Day
1	/	0
2	/	0
3	Leave a worker idle	0
4	Assign a worker to the new factory	100
5	Reassign workers to factories 2 and 100	102
6	Reassign workers to factories 2 and 100, leaving a worker idle	102
7	Assign a worker to the new factory	10102
8	Reassign workers to all factories	10103

The total profit is $0 + 0 + 0 + 100 + 102 + 102 + 10102 + 10103 = 20509$.

Problem E. Card Checking

Time limit: 2 seconds
Memory limit: 1024 megabytes

You're going to solve a problem related to the poker game "Landlords".

Game Introduction

"Landlords" is a card game where three players take turns playing cards. Let the three players be A, B, and C, respectively. Each player holds some cards in their hand, and the game continues until one player has played all of their cards.

Players A and B belong to the peasant team. If either A or B plays all of their cards, the peasants win. Player C alone forms the landlord team, and if C plays all of their cards, the landlord wins.

Playing Rules

In this problem, each card has an integer in $[1, n]$ written on it, called the rank of the card. The ranks are in increasing order: $1, 2, \ldots, n$.

Only the following two types of card combinations are allowed in this problem:

- Single: Any single card.
- Double: Two cards of the same rank.

Unlike in the usual Landlords game, in this problem, a player may play a single card only if he/she has exactly one card of that rank in hand. In other words, a pair in hand cannot be split into two single cards.

In each round of play, one player will be the starting player of that round. The starting player acts first and can play any legal card combination. Then, starting from the next player, players take turns in order. On each player's turn, they can make one of the following two decisions:

- Play a card combination that is of the same type as the last played cards in the current round and has a strictly higher rank.
- Choose to pass.

After a player plays a combination, if the other two players both choose to pass in succession, the current round ends. At this point, the player who made the last successful play becomes the starting player of the next round. Then a new round starts.

Problem

In this problem, consider the following situation:

- The landlord, C, has only one card left, with rank p_c.

- The hands of the two peasants, A and B, are described by two strings s_a and s_b of length n, consisting only of the characters `0`, `1`, and `2`. Here, the i-th character indicates how many cards of rank i the corresponding player holds.

All players' hands are public information; that is, every player knows the exact cards held by all players when making decisions. Player A is the starting player of the first round, and the three players always take turns in the order A $\to$ B $\to$ C $\to$ A $\to \cdots$. You need to determine whether the peasant team can win, assuming all players adopt optimal strategies.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^5$), indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 2.5 \times 10^5$).

The second line contains a string s_a of length n, consisting only of `0`, `1`, and `2`, indicating peasant A's hand.

The third line contains a string s_b of length n, consisting only of `0`, `1`, and `2`, indicating peasant B's hand.

The fourth line contains an integer p_c ($1 \le p_c \le n$), indicating the rank of landlord C's only card.

It is guaranteed that each player has at least one card.

It is also guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, output one line. If the peasant team can win, output `Yes`; otherwise, output `No`.

Example

standard input	standard output
4 9 001110201 002110211 7 9 110200000 222000000 7 9 222000000 110000000 7 9 111000002 111000210 7	Yes No Yes No

Note

We explain the first sample test case below.

#	**A**	**B**	**C**	**Play**
1	3,4,5,7,7,9	3,3,4,5,7,7,8,9	7	A (3) → B (8) → C (pass) → A (9) → B (pass) → C (pass)
2	4,5,7,7	3,3,4,5,7,7,9	7	A (4) → B (9) → C (pass) → A (pass)
3	5,7,7	3,3,4,5,7,7	7	B (3,3) → C (pass) → A (7,7) → B (pass) → C (pass)
4	5	4,5,7,7	7	A (5)

Problem F. Astra

Time limit: 1 second
Memory limit: 1024 megabytes

"Astra" is a beautiful boardgame where players take turns discovering constellations by marking the stars. This game is a clever mix of tactics and strategy, with a streamlined and intuitive rule set that makes it easy to pick up and quick to play.

Photo by @kavics004 on BoardGameGeek

Consider a two-player game between Alice and Bob. In this game, there is a tree-shaped constellation, which can be regarded as a connected graph consisting of n stars (numbered from 1 to n) and $(n-1)$ edges. At the beginning, only one star s is marked. Alice and Bob take turns marking the remaining stars according to the following rule:

- Each turn, a player should mark at least 1 star and at most k stars.
- Each star can be marked only once.
- Let $a_1, a_2, \cdots, a_p$ be the stars marked in a turn, in the order they are marked:
  - When marking a_1, it must be a neighbor of a marked star.
  - For all $2 \le i \le p$, when marking a_i, it must be a neighbor of a_{i-1}.

Two stars u and v are neighbors if they are directly connected by an edge.

The player who marks the last star wins the constellation. Given that Alice is the first to play and both players play optimally, for each $1 \le s \le n$, determine who will win the constellation.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 100$), indicating the number of test cases. For each test case:

The first line contains two integers n and k ($2 \le n \le 2 \times 10^3$, $1 \le k < n$), indicating the number of stars and the maximum number of stars marked in each turn.

For the following $(n-1)$ lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$), indicating that there is an edge connecting stars u_i and v_i.

It is guaranteed that the sum of n over all test cases does not exceed 2×10^3.

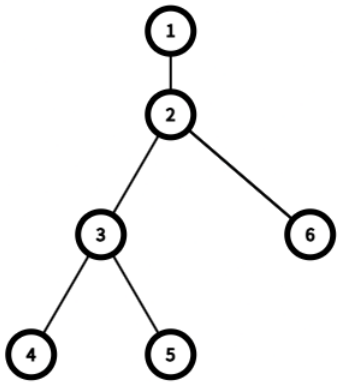
Output

For each test case, output one line containing a string $d_1 d_2 \cdots d_n$ ($d_i \in \{0, 1\}$), where d_i indicates who will win the constellation when $s = i$. If Alice will win, then $d_i = 1$; otherwise, if Bob will win, then $d_i = 0$.

Example

standard input	standard output
3 6 2 1 2 3 2 3 4 5 3 2 6 3 1 1 2 2 3 3 2 1 2 2 3	011000 000 101

Note



The first sample test case is illustrated above.

- Consider $s = 1$. For the first turn, Alice has three choices: mark $\{2\}$, $\{2, 3\}$, or $\{2, 6\}$. If Alice chooses $\{2, 6\}$, Bob can then mark $\{3\}$, and 2 separated stars will remain. For the following turns, each player can only mark one star, and Bob will be the winner. Alice's other choices can be analyzed likewise.
- Consider $s = 2$. For the first turn, Alice has five choices: mark $\{1\}$, $\{3\}$, $\{6\}$, $\{3, 4\}$, or $\{3, 5\}$. Alice can choose to mark $\{3\}$, and 4 separated stars will remain. For the following turns, each player can only mark one star, and Alice will be the winner.

For the second sample test case, as $k = 1$, there will be exactly $3 - 1 = 2$ turns, so Bob will always be the winner.

For the third sample test case, if $s = 1$ or $s = 3$, Alice can directly mark the remaining 2 stars and win the constellation.

Problem G. Snake

Time limit: 1 second
Memory limit: 1024 megabytes

This is an interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

You are playing the classic snake game on a grid of size $n \times m$. The rows are numbered 1 to n from top to bottom, and the columns are numbered 1 to m from left to right. We denote the cell at row i and column j as (i, j).

The snake can be represented as a sequence of coordinate pairs that determine where its body is located: $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$. Here, k denotes the length of the snake. The head of the snake is at (x_1, y_1), the tail is at (x_k, y_k), and neighboring parts of the body are located in cells that share a side.

Initially, the snake has length 1 and is located at a given cell (r_s, c_s).

There are 4 types of commands to move the snake:

- `U`: Command the snake to move one step up. The head will then move to $(x_1 - 1, y_1)$.
- `D`: Command the snake to move one step down. The head will then move to $(x_1 + 1, y_1)$.
- `L`: Command the snake to move one step left. The head will then move to $(x_1, y_1 - 1)$.
- `R`: Command the snake to move one step right. The head will then move to $(x_1, y_1 + 1)$.

When the head moves, each part of the body also moves accordingly. Specifically, the i-th part of the body ($2 \le i \le k$) moves to the position where the $(i-1)$-st part was before the command. Meanwhile, the cell previously occupied by the tail becomes empty.

The snake cannot move outside the grid. Besides, the snake cannot collide with itself — you must guarantee that no two parts of the body share the same cell after any command. Consider the following corner case: the head is at (x_1, y_1), and the tail is at (x_k, y_k). If the head is moving to (x'_1, y'_1), then it is allowed that $(x'_1, y'_1) = (x_k, y_k)$: if we think about a real-world scenario, the head moves into the cell just as the tail moves out. In a similar fashion, it is allowed to swap the head and the tail by using a single command when $k = 2$.

There are $(nm - 1)$ apples that will appear one at a time. Each time, an apple appears at some cell (r, c) that is not currently occupied by the snake. You need to output a sequence of commands that moves the snake's head to the apple. The head must arrive at (r, c) exactly on the last command, and it must not enter (r, c) before that.

For each sequence of commands you output, every command except the last one is a normal move: the tail vacates its cell as described above. The last command in the sequence is an eating move: the snake eats the apple when the head arrives at (r, c). On this move, the tail does NOT vacate its cell, so the length of the snake increases by 1.

Your task is to successfully eat all $(nm - 1)$ apples.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 100$), indicating the number of test cases. For each test case:

The first line contains four integers n, m, r_s, and c_s ($2 \le n, m \le 50$, $nm \le 100$, $1 \le r_s \le n$, $1 \le c_s \le m$), indicating the size of the grid and the starting position of the snake.

Interaction Protocol

You need to eat all $(nm - 1)$ apples one by one. For each apple:

- Read a line containing two integers r and c ($1 \le r \le n$, $1 \le c \le m$), indicating the position of the apple. It is guaranteed that the position of the apple is not currently occupied by the snake.
- Output a string consisting of the characters `U`, `D`, `L`, `R` — the sequence of commands that moves the snake's head from its current position to (r, c). To make this problem harder, we constrain the length of the string to be at most nm.

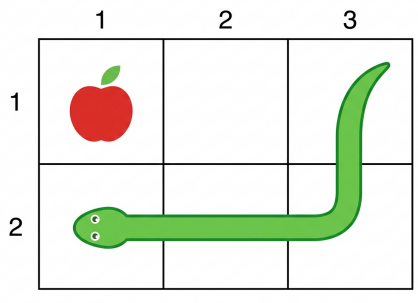
After printing each line, make sure to flush the output. It can be shown that an answer always exists.

Example

standard input	standard output
2 2 3 1 2 2 1 1 3 2 1 1 1 1 2 2 2 1 1 2 2 1 1 1 2	 LD URR DLL U R RD LU R

Note

The following image illustrates the first sample test case after eating 3 apples.



Problem H. Telepathy

Time limit:	3 seconds
Memory limit:	1024 megabytes

This is a multi-pass problem.

Alice and Bob are performing a telepathy trick for Christina. Christina chooses a subset $S \subset \{1, 2, \ldots, n\}$ such that $|S| < \frac{n}{2}$. Then, Alice adds a single integer x to this subset, such that $x \in \{1, 2, \ldots, n\}$ and $x \notin S$. Bob is only given the resulting set after Alice's addition and needs to deduce the exact integer Alice added.

Interaction Protocol

Your solution is executed twice on each test. In the first run, your solution acts as Alice. In the second run, your solution acts as Bob.

There are multiple test cases in each run.

First Run

The first line of the input contains the string `Alice`. The second line contains an integer T ($1 \le T \le 5 \times 10^5$), indicating the number of test cases. For each test case:

The first line of input contains two integers n and k ($3 \le n \le 10^6$, $1 \le k < \frac{n}{2}$). The second line contains k distinct integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n$), indicating the subset chosen by Christina.

For each test case, your solution should output one line containing $(k+1)$ distinct integers $b_1, b_2, \ldots, b_{k+1}$ ($1 \le b_i \le n$), such that $\{a_1, a_2, \ldots, a_k\} \subset \{b_1, b_2, \ldots, b_{k+1}\}$.

Second Run

Your solution will be restarted for the second run.

The first line of the input contains the string `Bob`. The second line contains an integer T ($1 \le T \le 5 \times 10^5$), indicating the number of test cases. For each test case:

The first line of input contains two integers n and k ($3 \le n \le 10^6$, $1 \le k < \frac{n}{2}$). The second line contains $(k+1)$ distinct integers $b_1, b_2, \ldots, b_{k+1}$ ($1 \le b_i \le n$), which is exactly the set you output for this test case in the first run (possibly in a different order).

For each test case, your solution should output one line containing a single integer, indicating the integer you added in the first run.

Note that the order of test cases in the second run might be different from that in the first run. Your solution is considered correct if you correctly deduce the added integer for all test cases.

It is guaranteed that the sum of n over all test cases does not exceed 3×10^6.

Examples

standard input	standard output
Alice 2 7 3 3 1 5 5 1 2	5 6 1 3 3 2
Bob 2 5 1 3 2 7 3 3 5 1 6	3 6

Note

The example shows two runs of a certain solution on the sample test cases.

In the first run, for the first sample test case, Christina has chosen the subset $S = \{1, 3, 5\}$ (given as `3 1 5`) from $\{1, 2, \ldots, 7\}$, with $|S| = 3 < \frac{7}{2}$. Alice must add one extra integer not in S and output the resulting set of size 4. She chooses to add the integer 6, so she outputs the set $\{1, 3, 5, 6\}$ (in the order `5 6 1 3`).

In the second run, the second sample test case corresponds to the first sample test case in the first run. Bob receives $(n, k) = (7, 3)$ and the set $\{1, 3, 5, 6\}$ (given as `3 5 1 6`). Using the pre-agreed strategy between Alice and Bob, he can deduce that 6 is the added integer and outputs `6`.

Problem I. Calendar Cubes

Time limit: 1 second
Memory limit: 1024 megabytes

Colin is designing a special calendar using two cubes.

Each cube has exactly 6 faces, and each face is labeled with a digit from 0 to 8. Using the two cubes, Colin wants to display dates in two-digit format.

For example, the illustration above shows two cubes displaying the integer 16 on the front. One cube shows the digit 1, and the other shows the digit 6.

To display an integer $d \in [01, 99]$ in two-digit format, choose one cube to show the tens digit (for 01 to 09, the tens digit is 0), and the other should show the units digit. Note that the digit 6 can also represent 9. That is, if a cube has a 6, it can be used as either 6 or 9.

Given two cubes, we define their MEX as the smallest **positive integer** in two-digit format that cannot be displayed. It can be shown that no two cubes can display all integers from 01 to 99 (both inclusive).

Given an integer x (given in two-digit format), you need to construct two cubes such that their MEX is exactly x.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 99$), indicating the number of test cases. For each test case:

The only line contains an integer x ($01 \le x \le 99$) in two-digit format.

Output

For each test case:

- If it is possible to construct two cubes such that their MEX is exactly x, first output `Yes` in one line. Then output a second line containing 12 integers (from 0 to 8) separated by a space, indicating the two cubes. The first 6 integers are the digits on the first cube, while the last 6 integers are the digits on the second cube.
- Otherwise, if it is impossible to do so, just output `No` in one line.

Example

standard input	standard output
4 01 02 99 11	Yes 0 0 0 0 0 0 0 0 0 0 0 0 Yes 1 1 1 1 1 1 0 0 0 0 0 0 No Yes 4 0 5 7 6 8 0 2 3 1 0 0

Problem J. Crossroads

Time limit: 3 seconds
Memory limit: 1024 megabytes

Given n vertical lines, where the i-th line is $x = a_i$, and also m horizontal lines, where the i-th line is $y = b_i$, it costs t_i units of time to move 1 unit of distance along the i-th vertical line, and it costs t_0 units of time to move 1 unit of distance along any horizontal line.

Let $f(i, j)$ be the minimum time needed to get from $(0, 0)$ to (a_i, b_j) by moving only along the vertical and horizontal lines. You need to answer q queries, where each query is represented as four integers l, r, d, and u, and you need to calculate

$$\sum_{i=l}^{r}\sum_{j=d}^{u} f(i, j)$$

Since the answers may be large, output them modulo 998 244 353.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The first line contains three integers n, m, and q ($1 \le n, m, q \le 2 \times 10^5$), indicating the numbers of vertical lines, horizontal lines, and queries.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 = a_1 < a_2 < \cdots < a_n \le 10^9$), indicating the x-coordinates of the vertical lines.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($0 = b_1 < b_2 < \cdots < b_m \le 10^9$), indicating the y-coordinates of the horizontal lines.

The fourth line contains n integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le 10^9$), where t_i indicates the time needed to move 1 unit of distance along the i-th vertical line.

The fifth line contains one integer t_0 ($1 \le t_0 \le 10^9$), indicating the time needed to move 1 unit of distance along any horizontal line.

For the following q lines, the i-th line contains four integers l_i, r_i, d_i, and u_i ($1 \le l_i \le r_i \le n$, $1 \le d_i \le u_i \le m$), indicating the i-th query.

It is guaranteed that the sum of n, the sum of m, and the sum of q over all test cases do not exceed 2×10^5.

Output

For each test case, output one line containing q integers, where the i-th integer is the answer to the i-th query modulo 998 244 353.

Example

standard input	standard output
1 3 2 3 0 2 10 0 3 100 2 1 1 1 3 1 2 2 3 2 2 1 1 1 1	43 21 0

Problem K. Sum and Product

Time limit: 1 second
Memory limit: 1024 megabytes

Let $f(n, k)$ be the number of integer pairs (a, b) such that:

- $0 \le a, b < n$.
- $k(a + b) \equiv ab \pmod{n}$.

Given two integers n and m, you need to calculate

$$\sum_{k=0}^{m} f(n, k)$$

Since the answer may be large, output it modulo 998 244 353.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The only line contains two integers n and m ($1 \le n \le 10^{14}$, $0 \le m < n$).

It is guaranteed that there are at most 10 test cases with n greater than 10^8.

Output

For each test case, output one line containing an integer, indicating the answer modulo 998 244 353.

Example

standard input	standard output
4 7 2 100 15 202604 11 1145141 919810	25 2500 3444208 165721266

Problem L. Critical Strike

Time limit: 1 second
Memory limit: 1024 megabytes

In a popular action game, there are n different critical strike equipments available in the shop. The i-th equipment has a probability of $\frac{p_i}{100}$ to trigger a critical strike with multiplier v_i, and costs w_i coins. Each equipment can be bought only once.

When multiple equipments trigger critical strikes simultaneously, only the highest multiplier takes effect. If no equipment triggers a critical strike, the effective multiplier is regarded as 0.

You can spend at most m coins to buy some equipments. You need to maximize the expected effective multiplier of the equipments you bought.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 300$), indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 3 \times 10^3$), indicating the number of equipments and the maximum number of coins you can spend.

For the following n lines, the i-th line contains three integers p_i, v_i, and w_i ($1 \le p_i \le 100$, $1 \le v_i \le 10^9$, $1 \le w_i \le 3 \times 10^3$), indicating the i-th equipment.

It is guaranteed that the sum of n and the sum of m over all test cases do not exceed 3×10^3.

Output

For each test case, output one line containing a single real number, indicating the maximum expected effective multiplier.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your answer is a and the jury's answer is b, your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Example

standard input	standard output
2 3 15 75 50 5 50 100 10 25 200 15 2 10 50 10 100 70 30 1000	68.750000000000 0.000000000000

Problem M. Night at the Museum

Time limit: 3 seconds
Memory limit: 1024 megabytes

A security guard patrols a museum to monitor exhibits. The patrol route is a closed polyline consisting of n points labeled from 1 to n, which are connected in order (the last point connects back to the first point) to form n consecutive segments. The guard patrols in the order $1 \to 2 \to \cdots \to n \to 1 \to \cdots$. When the guard is not at a segment endpoint, their field of view is a sector with radius r and central angle $2a$, with the sector's bisector aligned with the forward direction of the patrol segment.

A point P on a patrol segment (endpoints exclusive) can see an exhibit at point Q if and only if Q lies within the sector region with apex at P. Given the closed polyline patrol route defined by n points and m exhibit locations, for each $k = 0, 1, \ldots, m$, find the total length of patrol routes where exactly k exhibits can be seen.

Input

There is only one test case in each test file.

The first line contains four integers n, m, r, and a ($3 \le n \le 2 \times 10^3$, $1 \le m \le 2 \times 10^3$, $1 \le r \le 10^4$, $0 < a < 90$), indicating the number of points of the closed polyline, the number of exhibits, the radius of the field of view, and the half-angle of the field of view in degrees, respectively.

For the following n lines, the i-th line contains two integers x_i and y_i ($-10^4 \le x_i, y_i \le 10^4$), indicating the coordinates of the i-th point of the polyline. Consecutive points form patrol segments, and the last point connects back to the first point to form a closed loop. It is guaranteed that no two consecutive points (including the last and first points) are identical.

For the following m lines, the i-th line contains two integers x'_i and y'_i ($-10^4 \le x'_i, y'_i \le 10^4$), indicating the coordinates of the i-th exhibit.

Output

Output $(m + 1)$ lines, where the i-th line contains a real number, indicating the total length of patrol routes where exactly $(i - 1)$ exhibits can be seen.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your answer is a and the jury's answer is b, your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output
3 2 3 45 1 1 5 5 8 1 2 3 4 2	14.178145584873 2.247170915928 1.231537748691
4 1 3 60 0 0 6 6 6 0 0 6 2 1	25.279547784093 3.691014964384

Note

The first sample test case is illustrated below. A and B are the two exhibits, and G is the guard.