Problem A. Algebraic Teamwork

Source file name:	algebraic.c, algebraic.cpp, algebraic.java
Input:	`standard`
Output:	`standard`

The great pioneers of group theory and linear algebra want to cooperate and join their theories. In group theory, permutations – also known as bijective functions – play an important role. For a finite set A, a function $\sigma : A \to A$ is called a permutation of A if and only if there is some function $\rho : A \to A$ with

$$\sigma(\rho(a)) = a \text{ and } \rho(\sigma(a)) = a \text{ for all } a \in A.$$

The other half of the new team – the experts on linear algebra – deal a lot with idempotent functions. They appear as projections when computing shadows in 3D games or as closure operators like the transitive closure, just to name a few examples. A function $p : A \to A$ is called idempotent if and only if

$$p(p(a)) = p(a) \text{ for all } a \in A.$$

To continue with their joined research, they need your help. The team is interested in non-idempotent permutations of a given finite set A. As a first step, they discovered that the result only depends on the set's size. For a concrete size $1 \leq n \leq 10^5$, they want you to compute the number of permutations on a set of cardinality n that are **not** idempotent.

Input

The input starts with the number $t \leq 100$ of test cases. Then t lines follow, each containing the set's size $1 \leq n \leq 10^5$.

Output

Output one line for every test case containing the number modulo $1000000007 = (10^9 + 7)$ of **non**-idempotent permutations on a set of cardinality n.

Example

Input	Output
3 1 2 2171	0 1 6425

Problem B. Beam Me Out!

Source file name:	beam.c, beam.cpp, beam.java
Input:	`standard`
Output:	`standard`

King Remark, first of his name, is a benign ruler and every wrongdoer gets a second chance after repenting his crimes in the Great Maze!
Today's delinquent is a renowned computer scientist, but his fame didn't do him any good after he declined to do research on the so called and soon-to-be-famous Remark's algorithms! Those strange randomized algorithms may run indefinitely long (or even never terminate) and may or may not produce a right answer if terminated.
Handily, the Great Maze got recently a major upgrade with the newest beaming technology which made all doors obsolete: After the delinquent says the magic words "I was wrong and will never disappoint king Remark again!" he will be immediately beamed to the next room. It will be chosen randomly from a list of possible goal rooms.
The Great Maze consists of n rooms numbered 1 to n. Every detainee starts his quest for pardon in room 1 and hopes to get to the throne room n in which he will receive his pardon. If he ends up in a room, whose list of goal rooms is empty, his tour is over; through he could surely say the magic words again and again – that would not hurt, but would not help him either. Great king Remark, as most of the kings, doesn't like surprises and summoned you to answer two questions: Is it guaranteed, that the criminal will get to the throne room and is there a limit of beaming operations after which the game is over for sure.
You know better, than to disappoint the great king with a wrong answer or no answer at all, don't you?

Input

The input contains a single test case. It starts with a line consisting of an integer $2 \leq n \leq 50000$ – the number of rooms in the Great Maze. For each of the rooms 1 to $n-1$, two lines will follow representing the corresponding list of the goal rooms (in order 1 to $n-1$). Bear in mind, that after reaching the throne room n the quest is over. Thus, the list of the throne room is not a part of the input.
The first of these two lines will contain an integer $0 \leq m \leq n$ – the number of goal rooms on the list. The second line will contain a list of m goal rooms or an empty string, if m = 0. Each list will be sorted in strictly ascending order (this implies every number on the list will be unique) and consist from integers between 1 and n, inclusive.
The total number of goal rooms summed over all lists will not exceed 10^6.

Output

For each test case a line consisting of two words:

- the first word must be "`PARDON`", if the probability for the prisoner getting to the throne room during his random walk is 100%, or "`PRISON`" otherwise.
- the second word must be "`LIMITED`", if a limit for the number of beaming operations exists, or "`UNLIMITED`" otherwise.

Example

Input	Output
3 2 2 3 1 3	PARDON LIMITED
3 2 2 3 0	PRISON LIMITED
3 2 2 3 2 1 3	PARDON UNLIMITED
3 2 2 3 1 2	PRISON UNLIMITED

Problem C. Bounty Hunter

Source file name:	bounty.c, bounty.cpp, bounty.java
Input:	`standard`
Output:	`standard`

Spike is a bounty hunter and he is currently tracking a criminal! To investigate he uses his spaceship, the Swordfish II, and travels to N different places on 2D Euclidean space before returning to his crew at the starting location with all the information he has gathered. The starting location is the leftmost place (with the lowest x-coordinate) and Spike wants to travel to *every* other place before returning. However space fuel costs a lot of Woolongs and Spike would rather spend his money on special beef with bell peppers. Therefore he wants to travel the minimum possible distance. On top of that he is being chased by the Red Dragon crime syndicate. To make sure they don't catch him he can only visit places in increasing order of their x-coordinate until he reaches the rightmost place (with the largest x-coordinate), then he can turn around and visit places in decreasing order of their x-coordinate until he reaches his starting location again.

Input

The input starts with an integer T ($1 \leq T \leq 100$) specifying the number of test cases that follow. Each test case consists of an integer N ($2 \leq N \leq 512$) specifying the number of places in the tour. The coordinates of these places are given as integers in the next N lines, x-coordinate first, y-coordinate second ($0 \leq x, y \leq 5000$). The places are given in ascending order of the x-coordinate. Every place has a unique x-coordinate.

Output

For each test case, output on a single line the minimum travel distance needed to complete the tour. Your output should have an absolute or relative error of at most 10^{-2} .

Example

Input	Output
2 5 0 1 1 2 2 0 3 2 4 1 3 100 1 200 1 300 1	9.300563079746 400

Problem D. My brother's diary

Source file name:	diary.c, diary.cpp, diary.java
Input:	`standard`
Output:	`standard`

Nowadays, people who want to communicate in a secure way use asymmetric encryption algorithms such as RSA. However, my older brother uses another, simpler encryption method for his diary entries. He uses a substitution cipher where each letter in the plaintext is substituted by another letter from the alphabet. The distance between the plaintext letter and the encrypted letter is fixed. If we would define this fixed distance d to 5, A would be replaced by F, B by G, C by H, . . ., Y by D, Z by E.

With a fixed and known distance d the decryption would be somewhat simple. But my brother uses random distances for each of his diary entries. To decrypt his diary I have to guess the distance d for each entry. Thus, I use the well known phenomenon that the letter E is used more often in English words than other letters.

Can you write a program for me that calculates the distance d based on the fact that the most used letter in the encrypted text corresponds to the letter E in plaintext? Of course, I am interested in the decrypted text, too.

Input

The input consists of several test cases c that follow ($1 \leq c \leq 100$). Each test case is given in exactly one line containing one diary entry. Diary entries only use upper case letters ($A - Z$) and spaces. Each diary entry consists of at most 1000 encrypted letters (including spaces).

Output

For each test case, print one line containing the smallest possible distance d ($0 \leq d \leq 25$) and the decrypted text. If the decryption is not possible because there are multiple distances conforming to the rules above, print "NOT POSSIBLE" instead. Spaces are not encrypted.

Example

Input

```
4
RD TQIJW GWTYMJWX INFWD JSYWNJX ZXJ F XNRUQJ JSHWDUYNTS YJHMSNVZJ
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
XVIDRE TFCCVXZRKV GIFXIRDDZEX TFEKVJK UVTIPGKZFE
XVIDRE TFCCVXZRKV GIFXIRDDZEX TFEKVJK
```

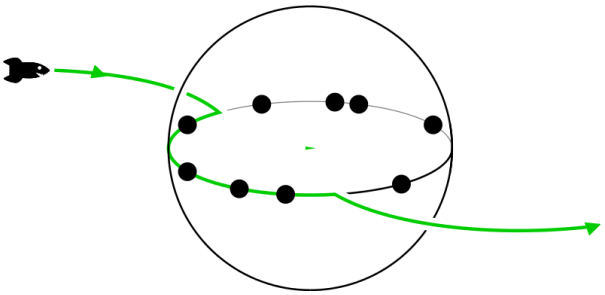
Output

```
5 MY OLDER BROTHERS DIARY ENTRIES USE A SIMPLE ENCRYPTION TECHNIQUE
10 JXU GKYSA RHEMD VEN ZKCFI ELUH JXU BQPO TEW
17 GERMAN COLLEGIATE PROGRAMMING CONTEST DECRYPTION
NOT POSSIBLE
```

Problem E. Equator

Source file name:	equator.c, equator.cpp, equator.java
Input:	`standard`
Output:	`standard`

In a galaxy far away, the planet *Equator* is under attack! The evil gang *Galatic Criminal People Cooperation* is planning robberies in Equator's cities. Your help is needed! In order to complete your training for becoming a lord of the dark side you should help them deciding which cities to rob.
As the name says, the desert planet Equator only can be inhabited on its equator. So the gang lands there at some point and travels into some direction robbing all cities on their way until leaving the planet again.



But what is still open for them is to decide where to land, which direction to take, and when to leave. Maybe they shouldn't even enter the planet at all? They do not consider costs for traveling or for running their ship, those are peanuts compared to the money made by robbery!
The cities differ in value: some are richer, some are poorer, some have better safety functions. So the gang assigned expected profits or losses to the cities. Help them deciding where to begin and where to end their robbery to maximize the money in total when robbing **every** city in between.

Input

The input starts with the number of test cases $T \leq 30$. Each test case starts a new line containing the number of cities $1 \leq n \leq 1000000$. In the same line n integers c_i follow. Each c_i ($0 \leq i \leq n$, $-1000 \leq c_i \leq +1000$) describes the money obtained when robbing city i, a negative c_i describes the amount of money they would lose.

Output

For each test case print one integer describing the maximum money they can make in total.

Example

Input	Output
3 3 1 2 3 8 4 5 -1 -1 1 -1 -1 5 2 -1 -1	6 14 0

Problem F. Gold Rush

Source file name:	gold.c, gold.cpp, gold.java
Input:	`standard`
Output:	`standard`

Alice and Bob are on an adventure trip. Deep in the woods they discover a mysterious deep cave which they enter flutteringly. They find an old console with a giant bar of gold in it. On the bar, there is a number n. Both tried to carry the gold out the cave, but it was still to heavy for one of them.
Suddenly a little fairy appears in the corner of the cave and approaches Alice and Bob: "This gold is heavy. It weights 2^n femto-grams (10^{-15}) and n can reach 62."
Bob answered: "What luck! Alice's knapsack can carry up to a femto-grams and mine b femto-grams with $a + b = 2^n$." Alice interjected: "But how can we divide the gold?"
Fairy: "I can help you with a spell that can burst one piece of gold into two equally weighted ones. But for each single spell, the cave will be locked one additional day."
Alice consults with Bob to use the help of the fairy and take all of the gold. How long will they be trapped if they are clever?

Input

The input starts with the number $t \leq 1000$ of test cases. Then t lines follow, each describing a single test case consisting of three numbers n, a and b with $a, b \geq 1, a + b = 2^n$, and $1 \leq n \leq 62$.

Output

Output one line for every test case with the minimal number of days that Alice and Bob are locked in the cave.

Example

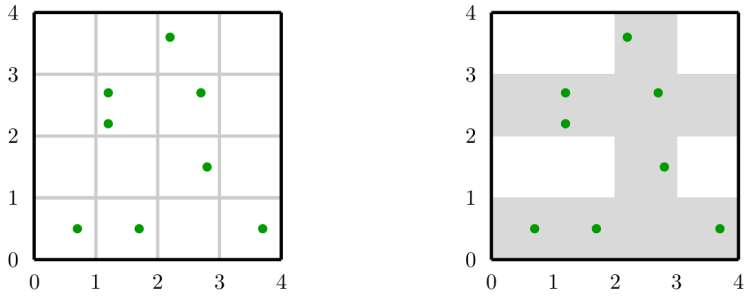
Input	Output
3 2 2 2 2 1 3 10 1000 24	1 2 7

Problem G. Jewelry Exhibition

Source file name:	jewelry.c, jewelry.cpp, jewelry.java
Input:	`standard`
Output:	`standard`

To guard the art jewelry exhibition at night, the security agency has decided to use a new laser beam system, consisting of sender-receiver pairs. Each pair generates a strip of light of one unit width and guards all objects located inside the strip. Your task is to help the agency and to compute for each exhibition room the minimum number of sender-receiver pairs which are sufficient to protect all exhibits inside the room.

Any room has a rectangle shape, so we describe it as an $[0, N] \times [0, M]$ rectangle in the plane. The objects we need to guard are represented as points inside that rectangle. Each sender is mounted on a wall and the corresponding receiver on the opposite wall in such a way that the generated strip is a rectangle of unit width and length either N or M. Since the new laser beam system is still not perfect, each sender-receiver pair can only be mounted to generate strips the corners of which have integer coordinates. An additional drawback is that the sender-receiver pairs can protect only items inside the strips, but not those lying on their borders. Thus, the security agency arranged the exhibits in such a way that both coordinates of any point representing an exhibit are non-integers. The figure below (left) illustrates eight items arranged in $[0, 4] \times [0, 4]$ (the second sample input). In the room, up to eight sender-receiver pairs can be mounted. The figure to the right shows an area protected by three sender-receiver pairs.



Input

The input starts with the number of exhibition rooms $R \leq 10$. Then the descriptions of the R rooms follow. A single description starts with a single line, containing three integers: $0 < N \leq 100$, $0 < M \leq 100$, specifying the size of the current room and $0 < K \leq 10^4$, for the number of exhibits. Next K lines follow, each of which consists of two real numbers x, y describing the exhibit coordinates. You can assume that $0 < x < N$, $0 < y < M$ and that x and y are non-integer.

Output

For every room output one line containing one integer, that is the minimum number of sender-receiver pairs sufficient to protect all exhibits inside the room.

Example

Input	Output
2 1 5 3 0.2 1.5 0.3 4.8 0.4 3.5 4 4 8 0.7 0.5 1.7 0.5 2.8 1.5 3.7 0.5 2.2 3.6 2.7 2.7 1.2 2.2 1.2 2.7	1 3

Problem H. JuQueen

Source file name:	juqueen.c, juqueen.cpp, juqueen.java
Input:	`standard`
Output:	`standard`

JuQueen is the super computer with the best performance allover Germany. It is on rank 8 in the famous top500 list with its 458 752 cores. It draws a lot of energy (up to 2 301 kW), so we want to reduce that by underclocking the unused cores.
The cluster scheduling algorithm which is in charge of distributing jobs over the nodes and cores of a cluster will issue the following speedstepping commands:

- change X S changes the frequency of core X by S steps
- groupchange A B S changes the frequency of every core in range [A,B] by S steps
- state X returns the current state of core X

To be safe for the future, your program should be able to handle 4 587 520 cores. The initial frequency for each core is 0.

Input

The input contains a single test case. It starts with a line containing three integers C, N, and O, where C is the number of cores ($1 \leq C \leq 4587520$) to manage, N is the number of frequency steps for each core ($1 \leq N \leq 10000$) and O is the number of operations in the test program ($1 \leq O \leq 50000$). Then follow O lines, each containing one command as described above. X, A and B are 0-based IDs of the cores ($0 \leq A, B, X < C; A \leq B$). S is an integer number of steps, possibly negative ($-N \leq S \leq +N$).
Both, the *change* and the *groupchange* command will increase (or decrease) in single steps and stop as soon as **one** core in the group reaches the minimal (0) or maximal frequency (N).

Output

Output one line for every operation in the input. For *change* and *groupchange* print the changed number of steps, for state print the current state.

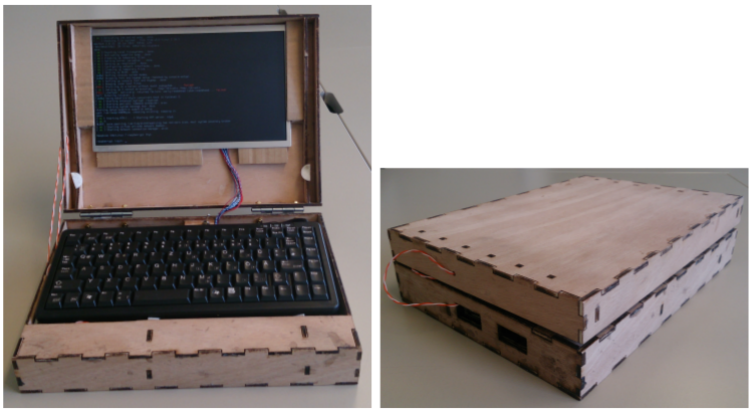
Example

Input	Output
10 10 5 state 0 groupchange 2 9 7 state 9 groupchange 0 2 10 change 0 -5	0 7 7 3 -3
4587520 10000 5 groupchange 0 4587010 9950 groupchange 23 4587000 42 groupchange 4710 4587001 -1000 state 1234560 groupchange 6666 3060660 10000	9950 42 -1000 8992 1008

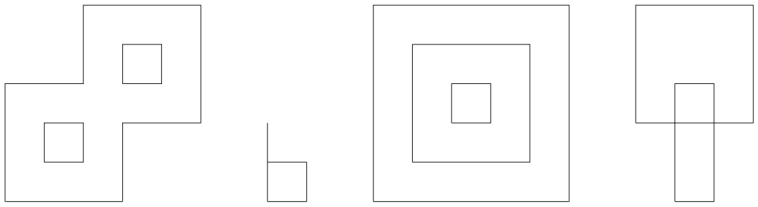
Problem I. Laser Cutting

Source file name:	laser.c, laser.cpp, laser.java
Input:	`standard`
Output:	`standard`

Jakob's laptop broke down, so he decided to build a new one by himself. After getting all the electronic parts, he needs to build a case. He decides to build one out of plywood, cut into pieces and glued together. He learns that, using the laser cutter of the FAU FABLAB, he can cut plywood sheets simply by creating a vector drawing. After some work, he has made a vector drawing. However, his drawing and calculating skills are rather poor, so he wants you to check his drawing.



The drawing consists of a number of polylines that are supposed to be non-self-intersecting polygons (which you have to check). To make things easier, all the lines that make up the polylines are parallel to either the x- or the y axis. Also, two polygons should not touch or intersect. Lastly, one part may contain some holes for plugging in other parts or electronics. However, a hole does not contain other parts or other holes. As parts and holes are both described by polygons, this means that any one polygon may be inside one other polygon, but not inside two other polygons.



Input

The input starts with the number of test cases t (with $t \leq 10$), on a line. Each test case starts with the number of polylines p (with $p < 50$), on a line. Every polyline starts with the number of coordinates n, with $5 \leq n \leq 50$. The next line contains $2n$ numbers, the x and y coordinates of the points on the polyline. You may assume that, given any two neighbouring points of a polyline, exactly one of their coordinates differ, i.e. all parts of a polyline are parallel to either x or y axis, and have non-zero length. The coordinates are integers between 0 and 10^6 .

Output

For each test case: If a polyline does not intersect with itself only on its first and last point, print "INVALID POLYGON". Else, if two polygons touch or intersect, print "INTERSECTING POLYGONS".

Else, if polygons are nested too deeply, print "INVALID NESTING". Lastly, if the description is correct, print "CORRECT".

Example

Input	Output
4 3 9 0 0 0 3 2 3 2 5 5 5 5 2 3 2 3 0 0 0 5 1 1 1 2 2 2 2 1 1 1 5 3 3 3 4 4 4 4 3 3 3 1 6 0 0 0 2 0 1 1 1 1 0 0 0 3 5 2 2 2 3 3 3 3 2 2 2 5 1 1 1 4 4 4 4 1 1 1 5 0 0 0 5 5 5 5 0 0 0 2 5 1 0 2 0 2 3 1 3 1 0 5 0 2 3 2 3 5 0 5 0 2	CORRECT INVALID POLYGON INVALID NESTING INTERSECTING POLYGONS

Problem J. Not a subsequence

Source file name:	subsequence.c, subsequence.cpp, subsequence.java
Input:	`standard`
Output:	`standard`

In this problem we consider strings over a fixed finite alphabet of size k. The alphabet contains the first k characters from the list

$$a, b, c, \ldots, z, A, B, C, \ldots, Z, 0, 1, \ldots, 9.$$

For every test case, we are given the value of k (notice that it cannot exceed 62), and consider only strings consisting of the first k characters from the list.
Given a string $s[1..n]$, we are interested in strings which are **not** its subsequences. Formally, a string $t[1..m]$ is a subsequence of a string $s[1..n]$ when one can choose **not necessarily contiguous** indices $1 \leq i_1 < i_2 < \ldots i_m \leq n$ such that $t[1] = s[i_1], t[2] = s[i_2], \ldots, t[m] = t[i_m]$. For example, *acb* is a subsequence of *babcaab*. Now, given a string $s[1..n]$, we would like to compute the smallest m such that there is a string $t[1..m]$, which is **not** a subsequence of $s[1..n]$. Additionally, we would like to count the number of such shortest strings $t[1..m]$. As the latter number can be quite large, output it modulo 10^9+7.

Input

The input starts with the number of test cases $T \leq 100$. Then the descriptions of T test cases follow. A single test case consists of a single line containing the size of the alphabet $k(k \in [1, 62])$ and the string $s[1..n](n \in [1, 10^6])$. The string consists of the first k characters from $a-z$ $A-Z$ $0-9$.

Output

For every test case output one line containing two numbers. The first number is the smallest m such that there is a string $t[1..m]$ consisting of the first k characters from $a-z$ $A-Z$ $0-9$, which is not a subsequence of $s[1..n]$. The second number is the total count of such shortest strings $t[1..m]$ modulo 10^9+7.

Example

Input	Output
3 2 abba 62 0123456789 3 aabbcbbcbabcbab	3 5 1 52 4 7

Problem K. Pizza voting

Source file name:	pizza.c, pizza.cpp, pizza.java
Input:	`standard`
Output:	`standard`

You are training for a programming contest with your team members Alice and Bob. After some hours of hard training you want to have a break and eat pizza. You decided to order a big pizza for all three of you. But you have to choose the kind of pizza you want to eat.
You know your favorite kind. But Alice and Bob have other constraints: Alice is on a diet so she wants a pizza with less calories as possible. Bob is just mean to Alice so he votes for as much caloriesas possible.

You decide to vote on which kind of pizza you order. As voting for one pizza wouldn't lead anywhere, you decide to use a veto voting. So everyone of you veto on pizza in a round robin manner. First Alice vetos one pizza, then Bob vetos one, at last you are allowed to veto. Then Alice has the next veto again, then Bob etc. until only one pizza is left.
Reminder: Alice will always veto the pizza with the most calories. Bob vetos always the pizza with least calories. You try to be clever in such a way that your favorite pizza is the remaining pizza.

Input

The input starts with the number of pizzas $n(1 \leq n \leq 100000)$ and the index of your favorite pizza $p(1 \leq p \leq n)$ (1-indexed). Then follow the description of the n pizzas, each given in one line. The description consists of one integer $c(0 \leq c \leq 1000000)$ giving the calories followed by a single word w giving the name of the pizza (up to 100 characters). The pizzas are ordered from low calories to high calories and the number of calories is unique for every pizza.

Output

Output one line. "YES" if you can vote in a way, such that your pizza will be selected. "NO" if you are not able to influence the vote in a way that your pizza will be selected.

Example

Input	Output
5 2 500 Margherita 600 Salami 700 Hawai 800 Speciale 900 Doener	YES
5 4 500 Margherita 600 Salami 700 Hawai 800 Speciale 900 Doener	NO