NWERC 2011

The 2011 ACM Northwestern Europe Programming Contest

acm International Collegiate Programming Contest

IBM event sponsor

The Problem Set

A Binomial coefficients

Gunnar is quite an old and forgetful researcher. Right now he is writing a paper on security in social networks and it actually involves some combinatorics. He wrote a program for calculating binomial coefficients to help him check some of his calculations.

A binomial coefficient is a number

$$\binom{n}{k} = \frac{n!}{k!(n-k)!},$$

where n and k are non-negative integers.

Gunnar used his program to calculate $\binom{n}{k}$ and got a number m as a result. Unfortunately, since he is forgetful, he forgot the numbers n and k he used as input. These two numbers were a result of a long calculation and they are written on one of many papers lying on his desk. Instead of trying to search for the papers, he tried to reconstruct the numbers n, k from the output he got. Can you help him and find all possible candidates?

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with an integer m ($2 \leq m \leq 10^{15}$): the output of Gunnar's program.

Output

Per test case:

- one line with an integer: the number of ways of expressing m as a binomial coefficient.
- one line with all pairs (n, k) that satisfy $\binom{n}{k} = m$. Order them in increasing order of n and, in case of a tie, order them in increasing order of k. Format them as in the sample output.

Sample in- and output

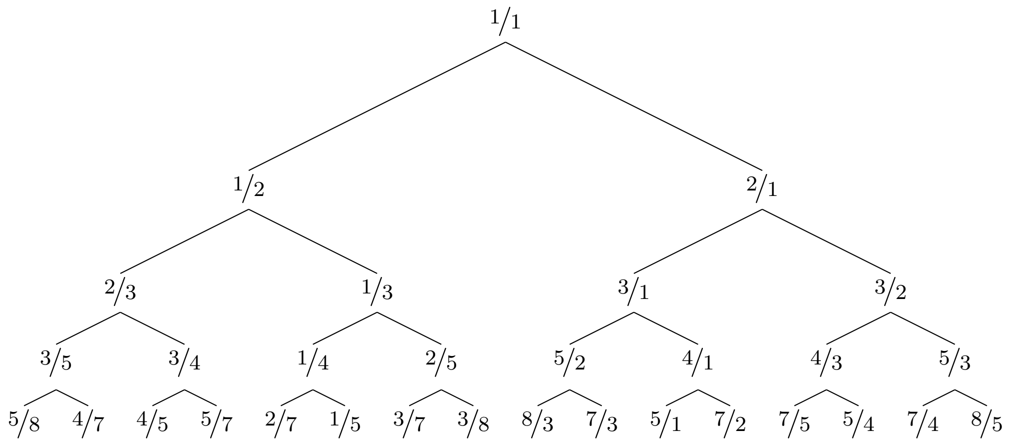
Input	Output

`2`
`2`
`15`
	`1`
	`(2,1)`
	`4`
	`(6,2) (6,4) (15,1) (15,14)`

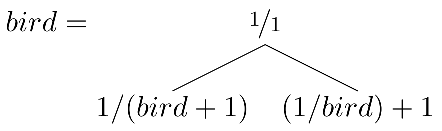
Almost blank page

B Bird tree

The Bird tree[1] is an infinite binary tree, whose first 5 levels look as follows:



It can be defined as follows:



This is a *co-recursive* definition in which both occurrences of $bird$ refer to the full (infinite) tree. The expression $bird + 1$ means that 1 is added to every fraction in the tree, and $1/bird$ means that every fraction in the tree is inverted (so a/b becomes b/a).

Surprisingly, the tree contains every positive rational number exactly once, so every reduced fraction is at a unique place in the tree. Hence, we can also describe a rational number by giving directions (`L` for left subtree, `R` for right subtree) in the Bird tree. For example, $2/5$ is represented by `LRR`. Given a reduced fraction, return a string consisting of `L`'s and `R`'s: the directions to locate this fraction from the top of the tree.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers a and b ($1 \leq a, b \leq 10^9$), separated by a `'/'`. These represent the numerator and denominator of a reduced fraction. The integers a and b are not both equal to 1, and they satisfy $\gcd(a, b) = 1$.

For every test case the length of the string with directions will be at most $10\,000$.

Output

Per test case:

- one line with the string representation of the location of this fraction in the Bird tree.

[1]Hinze, R. (2009). The Bird tree. *J. Funct. Program.*, 19:491–508.

Sample in- and output

Input	Output

`3`
`1/2`
`2/5`
`7/3`
	`L`
	`LRR`
	`RLLR`

C Movie collection

Mr. K. I. has a very big movie collection. He has organized his collection in a big stack. Whenever he wants to watch one of the movies, he locates the movie in this stack and removes it carefully, ensuring that the stack doesn't fall over. After he finishes watching the movie, he places it at the top of the stack.

Since the stack of movies is so big, he needs to keep track of the position of each movie. It is sufficient to know for each movie how many movies are placed above it, since, with this information, its position in the stack can be calculated. Each movie is identified by a number printed on the movie box.

Your task is to implement a program which will keep track of the position of each movie. In particular, each time Mr. K. I. removes a movie box from the stack, your program should print the number of movies that were placed above it before it was removed.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers n and m ($1 \leq n, m \leq 100\,000$): the number of movies in the stack and the number of locate requests.
- one line with m integers $a_1, \ldots, a_m$ ($1 \leq a_i \leq n$) representing the identification numbers of movies that Mr. K. I. wants to watch.

For simplicity, assume the initial stack contains the movies with identification numbers $1, 2, \ldots, n$ in increasing order, where the movie box with label 1 is the top-most box.

Output

Per test case:

- one line with m integers, where the i-th integer gives the number of movie boxes above the box with label a_i, immediately before this box is removed from the stack.

Note that after each locate request a_i, the movie box with label a_i is placed at the top of the stack.

Sample in- and output

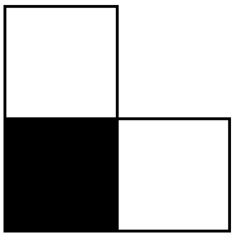
Input	Output

```
2
3 3
3 1 1
5 3
4 4 5
```
	```
	2 1 0
	3 0 4
	```

Almost blank page

D Piece it together

Tom has developed a special kind of puzzle: it involves a whole bunch of identical puzzle pieces. The pieces have the shape of three adjoint squares in an L-shape. The corner square is black, the two adjacent squares are white.



A puzzle piece

The puzzler is given a pattern of black and white squares in a rectangular grid. The challenge is to create that pattern using these pieces. The pieces can be rotated, but must not overlap.

Tom has already designed a few nice patterns, but he needs to find out if they can be constructed with the pieces at all. Rather than trying to test this for each pattern by hand, he wants to write a computer program to determine this for him. Can you help him?

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers n and m ($1 \leq n, m \leq 500$): the height and width of the grid containing the pattern, respectively.
- n lines, each containing m characters, denoting the grid. Each character is 'B', 'W', or '.', indicating a black, white or empty square respectively.

The grid contains at least one black or white square.

Output

Per test case:

- one line with either "YES" or "NO", indicating whether or not it is possible to construct the pattern with the puzzle pieces. You may assume that there is an infinite supply of pieces.

Sample in- and output

Input	Output
`2`	`YES`
`3 4`	`NO`
`BWW.`	
`WWBW`	
`..WB`	
`3 3`	
`W..`	
`BW.`	
`WBW`	

E Please, go first

You are currently on a skiing trip with a group of friends. In general, it is going well: you enjoy the skiing during the day and, of course, the après-skiing during the night. However, there is one nuisance: the skiing lift. As always, it is too small, and can only serve one person every 5 seconds. To make matters worse, you and your friends generally don't arrive simultaneously at the lift, which means that you spend time waiting at the bottom of the mountain for the lift and at the top again for your friends.

The waiting at the top is especially inefficient. In fact, you realize that if your friends haven't arrived yet, you might as well let other people pass you in the queue. For you, it makes no difference, since otherwise you'd be waiting at the top. On the other hand, your actions might save them time if their friends have already arrived and are currently waiting for them at the top.

You are wondering how much time would be saved if everybody adopts this nice attitude. You have carefully observed the queue for a while and noticed which persons form groups of friends. Suppose someone lets another pass if doing this doesn't change his own total waiting time, but saves time for the other person. Do this over and over again until it can't be done anymore. How much time will this save, in total?

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with an integer n $(1 \leq n \leq 25\,000)$: the number of people in the line for the lift.
- one line with n alphanumeric characters (uppercase and lowercase letters and numbers): the queue. The first person in this line corresponds to the person at the head of the queue. Equal characters correspond to persons from the same group of friends.

Output

Per test case:

- one line with an integer: the time saved, in seconds.

Sample in- and output

Input	Output
2 6 AABABB 10 Ab9AAb2bC2	15 45

Almost blank page

F Pool construction

You are working for the International Company for Pool Construction, a construction company which specializes in building swimming pools. A new client wants to build several new pool areas.

A pool area is a rectangular grid of $w \times h$ square patches, consisting of zero or more (possibly disconnected) pools. A pool consists of one or multiple connected hole patches, which will later be filled with water. In the beginning, you start with a piece of land where each patch is either a hole in the ground ('.') or flat grass ('#'). In order to transform this land into a pool area, you must adhere to the following:

- You can leave a patch as it is. This costs nothing.
- If the patch is grass in the beginning, you can dig a hole there. This costs d EUR.
- If the patch is a hole in the beginning, you can fill the hole and put grass on top. This costs f EUR.
- You *must* place special boundary elements along each edge running between a final grass patch and a final hole patch, to ensure that water does not leak from the pool. This costs b EUR per boundary element.
- The outermost rows and columns of the pool area must always be grass.

You are given the task of calculating the cost of the cheapest possible pool area given the layout of the existing piece of land.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers w and h ($2 \leq w, h \leq 50$): the width and height of the building site.
- one line with three integers d, f and b ($1 \leq d, f, b \leq 10\,000$): the costs for digging a new hole, filling an existing hole, and building a boundary element between a pool and grass patch.
- h lines of w characters each, denoting the layout of the original building site.

Output

Per test case:

- one line with an integer: the cost of building the cheapest possible pool area from the original piece of land.

Sample in- and output

Input

```
3
3 3
5 5 1
#.#
#.#
###
5 4
1 8 1
#..##
##.##
#.#.#
#####
2 2
27 11 11
#.
.#
```

Output

```
9
27
22
```

G Smoking gun

Andy: ”Billy the Kid fired first!”

Larry: ”No, I’m sure I heard the first shot coming from John!”

The arguments went back and forth during the trial after the big shoot-down, somewhere in the old wild west. Miraculously, everybody had survived (although there were serious injuries), but nobody could agree about the exact sequence of shots that had been fired. It was known that everybody had fired at most one shot, but everything had happened very fast. Determining the precise order of the shots was important for assigning guilt and penalties.

But then the sheriff, Willy the Wise, interrupted: ”Look, I’ve got a satellite image from the time of the shooting, showing exactly where everybody was located. As it turns out, Larry was located much closer to John than to Billy the Kid, while Andy was located just slightly closer to John than to Billy the Kid. Thus, because sound travels with a finite speed of 340 meters per second, Larry may have heard John’s shot first, even if Billy the Kid fired first. But, although Andy was closer to John than to Billy the Kid, he heard Billy the Kid’s shot first – so we know for a fact that Billy the Kid was the one who fired first!

Your task is to write a program to deduce the exact sequence of shots fired in situations like the above.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers n ($2 \leq n \leq 100$) and m ($1 \leq m \leq 1\,000$): the number of people involved and the number of observations.
- n lines with a string S, consisting of up to 20 lower and upper case letters, and two integers x and y ($0 \leq x, y \leq 1\,000\,000$): the unique identifier for a person and his/her position in Cartesian coordinates, in metres from the origin.
- m lines of the form “`S1 heard S2 firing before S3`”, where $S1$, $S2$ and $S3$ are identifiers among the people involved, and $S2 \neq S3$.

If a person was never mentioned as $S2$ or $S3$, then it can be assumed that this person never fired, and only acted as a witness. No two persons are located in the same position.

The test cases are constructed so that an error of less than 10^{-7} in one distance calculation will not affect the output.

Output

Per test case:

- one line with the ordering of the shooters that is compatible with all of the observations, formatted as the identifiers separated by single spaces.

If multiple distinct orderings are possible, output “`UNKNOWN`” instead. If no ordering is compatible with the observations, output “`IMPOSSIBLE`” instead.

Sample in- and output

Input

```
3
4 2
BillyTheKid 0 0
Andy 10 0
John 19 0
Larry 20 0
Andy heard BillyTheKid firing before John
Larry heard John firing before BillyTheKid
2 2
Andy 0 0
Beate 0 1
Andy heard Beate firing before Andy
Beate heard Andy firing before Beate
3 1
Andy 0 0
Beate 0 1
Charles 1 3
Beate heard Andy firing before Charles
```

Output

```
BillyTheKid John
IMPOSSIBLE
UNKNOWN
```

H Tichu

Tichu is a card game played by four players. The players sit around a square table, and each player forms a team with the person sitting opposite him or her. The game is played with a standard deck of cards and four additional special cards. The basic rule of the game is as follows: the player who won the last trick can start a new trick with any legal combination of cards. Then, in turn, each next player can either pass or play the same combination of cards, but with a higher value. This continues until everyone passes, and at that point the player who played the last combination wins the trick and can start a new trick. The main goal is to get rid of all of your cards as soon as possible.

These basic rules make it a good tactic to combine the cards in such a way that they can be played in as few combinations as possible. For simplicity we consider here a slightly modified version of the game. We ignore the special cards, so that leaves a standard deck of 52 cards, ranging over the values `2` to `Ace` and over the suits `hearts`, `diamonds`, `clubs`, and `spades`. The suits are indicated by the lowercase letters `h`, `d`, `c`, and `s`, while the values are indicated in increasing order by `2`–`9`, `T`, `J`, `Q`, `K`, `A`.

The following list is a complete set of legal combinations:[2]

- any single card;
- a pair of cards of the same value;
- three cards of the same value;
- four cards of the same value;
- a full house, that is, three cards of the same value and two cards of another, same value, for example `444KK`;
- a straight of length at least five, that is, at least five cards of consecutive increasing values, for example `89TJQK`.

In this problem, your task is to determine the minimum number of combinations that your hand of 13 cards can be partitioned into.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line that describes your hand of 13 cards. The card descriptions are separated by single spaces. A card is described by two characters: the value followed by the suit. All cards in your hand are different.

Output

Per test case:

- one line containing an integer n: the minimum number of combinations that your hand can be partitioned into.

[2]Those who know the game of Tichu might have noticed that we removed consecutive pairs of cards as a valid combination.

- n lines that describe a minimal set of combinations of cards from your hand. Each line should contain the cards in one legal combination, in the same format as in the input. All cards from your hand must occur exactly once in one of the combinations. No specific ordering of the combinations or the cards within a combination is required.

Sample in- and output

Input	Output
`2`	`4`
`2h 3c 4d 5d 6s Th Qc Qs Ad Tc Ts 9c 9d`	`2h 3c 4d 5d 6s`
`2h 3h 4h 5h 6d 7s 8h 8d 8c 8s 9c Td Js`	`Th Ts Tc Qc Qs`
	`9d 9c`
	`Ad`
	`2`
	`2h 3h 4h 5h 6d 7s 8d 9c Td Js`
	`8h 8s 8c`

I Tracking RFIDs

Jeroen operates a warehouse storage facility for the North Western Electrical Resource Company (NWERC). When a customer places an order with NWERC, this order is conveyed to the warehouse. Jeroen's task is to then find the products ordered, pack them into a box, and ship them to the customer.

NWERC has an unusual warehouse policy: the products are not arranged in any particular order, and are strewn all over the place. However, it is possible for Jeroen to do his job because each product is tracked using RFID technology[3]. Specifically, each product is assigned a wireless RFID chip as soon as it enters the warehouse, and sensors located on the warehouse ceiling are used to automatically track the products.

By default, each sensor has a range of r units – that is, it can read any RFID chip that is located at most r units from it in a straight line. However, if the line segment between a sensor and a product intersects with or touches x walls, the range of the sensor is reduced by x units in that direction. Furthermore, the sensors may fail to read an RFID chip due to interference from other sensors, so the distance between any pair of sensors in the warehouse is guaranteed to be at least r units. You may further assume that no sensor or product is placed on a wall.

Jeroen now wants to determine, for each product, which sensors can read its RFID chip. Can you help him?

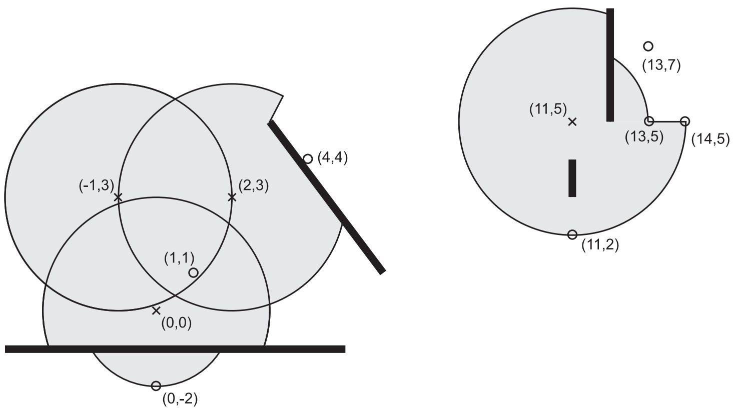


Illustration of sensors, walls and products as in the Sample Input.

Input

On the first line one positive integer: the number of test cases, at most 100. After that per test case:

- one line with four integers s ($1 \leq s \leq 250\,000$), r ($1 \leq r \leq 20$), w ($0 \leq w \leq 10$) and p ($1 \leq p \leq 10\,000$) where s represents the number of sensors, r the range of each sensor, w the number of walls and p the number of products.
- s lines containing two integers x_i and y_i ($-10\,000 \leq x_i, y_i \leq 10\,000$). Each such line represents a sensor at location (x_i, y_i). The distance between any pair of sensors is guaranteed to be at least r units.

[3]Objects, that have a radio-frequency identification (RFID) tag attached, can be tracked using radio waves.

- w lines containing four integers bx_i, by_i, ex_i and ey_i ($-10\,000 \leq bx_i, by_i, ex_i, ey_i \leq 10\,000$). Each such line represents a wall, which should be considered as straight line segment from (bx_i, by_i) to (ex_i, ey_i). The length of this line segment will be positive.
- p lines, each containing two integers px_i and py_i ($-10\,000 \leq px_i, py_i \leq 10\,000$). Each such line represents a product at location (px_i, py_i).

Output

Per test case:

- p lines, each representing a product in the order they appear in the input. Each line should contain an integer t, the number of sensors that can track the product; this integer should then be followed by a list of t ordered pairs representing the corresponding sensor locations. If there are multiple sensors, they should be sorted in increasing order by x-coordinate. If multiple sensors have the same x-coordinate, they should be sorted in increasing order by y-coordinate.

Sample in- and output

Input	Output

```
1
4 3 4 7
0 0
-1 3
2 3
11 5
-4 -1 5 -1
3 4 6 1
11 4 11 3
12 5 12 8
1 1
0 -2
4 4
11 2
13 5
13 7
14 5
```
	```
	3 (-1,3) (0,0) (2,3)
	1 (0,0)
	0
	0
	1 (11,5)
	0
	0
	```

J Train delays

Last year, some of the judges tried to travel to NWERC'10 by train. This turned into a big disaster: on the way there, a fire in a control room caused huge delays, while on the return trip, trains in Bremen were delayed due to a terrorist threat in Hamburg. Of course, these huge delays caused other delays in the train schedule, so the big question was which trains to take: would it be better to take this slow regional train now, or wait for that intercity train, which has a big chance of being delayed?

This year, the judges have planned ahead and carefully analyzed the train schedule. They even kept track of how often trains were delayed and by how much. Now that they have all this information, they want to travel as quickly possible, minimizing the expected duration of the journey. Can you help them?

For each train connection, the judges know exactly what its scheduled departure time and duration are, as well as the probability that its arrival at the destination will be delayed. You may assume that the probabilities of delays are independent and that the judges can adapt their itinerary as they go, depending on any delays which they might already have incurred. Trains always depart on time, but may arrive late and the judges do not know whether a train's arrival will be delayed until they have boarded it. It takes judges no time to switch trains, so they can take a connecting train that departs at the same time as they arrive at a place.

The judges can choose the time of their initial departure as they wish and they want to minimize the expected duration[4] of their total trip.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with the judges' place of origin and destination, these are different.
- one line with an integer n ($1 \leq n \leq 1\,000$): the number of train connections.
- n lines, each describing a train connection:
  - the origin and destination of this connection, these are different.
  - an integer m ($0 \leq m \leq 59$), the departure time in minutes after each full hour.
  - an integer t ($1 \leq t \leq 300$), the standard journey time (assuming no delays).
  - an integer p ($0 \leq p \leq 100$), the probability of delays as a percentage.
  - an integer d ($1 \leq d \leq 120$), the maximum delay in minutes.

All place names are given as strings of upper and lower case alphabetical characters, of length at most 20. If a train is delayed, then the length of the delay will be a whole number of minutes, and will be uniformly distributed in the range $[1, d]$.

[4] Given a travel plan of which trains to take (depending on previous connections and delays), the expected trip duration E is defined to be the sum of the trip duration T_i for each itinerary i possibly taken multiplied by the chance p_i of that itinerary occurring: $E = \sum_i p_i T_i$.

Output

Per test case:

- one line with a floating point number: the minimum expected duration of the total trip in minutes.

This number should be accurate up to 10^{-6} relative or absolute precision. Output `IMPOSSIBLE` instead if the destination is not reachable.

Sample in- and output

Input

```
3
Hamburg Bremen
3
Hamburg Bremen 15 68 10 5
Hamburg Bremen 46 55 50 60
Bremen Frankfurt 14 226 10 120
Amsterdam Rotterdam
1
Amsterdam Utrecht 10 22 5 10
BremenVegesack Utrecht
9
BremenVegesack BremenHbf 15 10 0 1
BremenVegesack BremenHbf 45 10 0 1
BremenVegesack Leer 23 140 10 15
BremenHbf Osnabruck 44 51 60 70
Osnabruck Amersfoort 55 147 38 40
Amersfoort Utrecht 24 15 30 15
Amersfoort Utrecht 54 15 10 35
Leer Groningen 45 140 5 10
Groningen Amersfoort 46 96 10 20
```

Output

```
68.3
IMPOSSIBLE
305.0532857
```

Note in the first example that it is better to take the slower train from Hamburg to Bremen, since the fast train would give an expected travel time of 70.25 minutes.