Benelux Algorithm Programming Contest

Preliminaries

Saturday October 1st 2011

Contents

A Stifling the Mutiny

Problem

A group of pirates is travelling in a convoy of ships, sailing in a row. However, the pirate captain is starting to lose control, and some disloyal pirates are ready to mute. As soon as on any ship S, the number of loyal pirates on S is outnumbered by the combined number of disloyal pirates on S, the previous ship (if S is not the first), and the next ship (if S is not the last) in the convoy, the disloyal pirates on these ships will row to S and take it over. To prevent an outbreak of mutiny, the captain decides to distribute the loyal and disloyal pirates over the ships in such a way that the disloyal pirates cannot capture any ship. Of course, each ship must have at least one loyal pirate to operate the ship.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers n and k, where $1 \leq n \leq 15$ and $n \leq k \leq 40$. The first number is the number of ships; the second number is the total number of pirates (whether loyal or disloyal) in the convoy.

Output

For every test case in the input, the output should contain one integer on a single line: the maximum number of disloyal pirates that the captain can distribute over the ships such that the disloyal pirates cannot capture any ship.

Example

Input	Output
3	1
1 3	1
3 4	5
3 16	

This page is intentionally left blank.

B RNG in Reverse

Problem

Tom is making a new computer game. It's an adventure game featuring both ninjas and pirates; fun guaranteed!
Various elements of the game contain a random factor. Therefore, the game makes use of various random number generators, or RNG for short. Each RNG uses the following formula: let x be the previous random number. The next number y is then given by:

$$y = ax^2 + bx + c \pmod{2^n}$$

where a, b, c and n are some integer parameters.
For the purpose of testing the game, Tom can interrupt the game and consult debug output. Some of the valuable pieces of information are the last values the RNGs have produced. However, Tom often wishes to backtrack his program, and for that he needs to figure out what the previous values of the RNGs were. So in short, given the parameters of a certain RNG and the current number y, he wants to know what the previous number x was. Can you help him?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with five integers y, a, b, c and n, satisfying $0 \leq y, a, b, c < 2^n$ and $1 \leq n \leq 31$: the last value of the RNG and its four parameters, respectively.

Output

For every test case in the input, the output should contain one integer on a single line: the previous value x of the RNG ($0 \leq x < 2^n$). If there is no such number, or more than one such number, the output should be "No unique solution" (without the quotation marks) on a single line.

Examples

Input	Output

4
26 2 1 5 5
10 1 0 0 4
1 1 1 1 4
3 14 15 92 7
	3
	No unique solution
	No unique solution
	55

This page is intentionally left blank.

C Attack of the Giant n-pus

Problem

A pirate ship is being attacked by a giant n-pus[1]. The n tentacles and the head of the creature have pierced the deck and are wreaking havoc on the ship. In an attempt to stop the creature from completely destroying the ship, the captain charges towards the head of the creature. Unfortunately, he quickly gets knocked back by one of the tentacles. The captain realizes he cannot attack the head of the n-pus as long as it can move its tentacles freely.
Luckily the captain is not alone on the ship. There are p ($p \geq n$) pirates spread around on the deck, ready to follow the captain's orders. So the captain comes up with the following plan. If each tentacle is attacked by one of the pirates, he can move freely towards the head of the creature and finish it off. To be safe, the captain will start moving only after each of the tentacles is being attacked by a pirate. When the captain reaches the head, he can instantly kill the creature. The captain wants to figure out which pirates to send to which tentacles, such that the creature can be killed as fast as possible. As this happens regularly to pirates, the captain wants you to write a program to solve this problem.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers n and p, satisfying $1 \leq n \leq p \leq 100$: the number of tentacles of the n-pus and the number of pirates (excluding the captain), respectively.
- One line with three integers x_c, y_c and v_c: the captain's coordinates and speed, respectively.
- p lines, each with three integers x_i, y_i and v_i: the coordinates and speed, respectively, of each pirate.
- One line with two integers x_h and y_h: the coordinates of the head of the n-pus.
- n lines, each with two integers x_j and y_j: the coordinates of each tentacle.

All coordinates satisfy $0 \leq x, y \leq 10,000$. All speeds satisfy $1 \leq v \leq 100$.
The captain, the pirates, the head and the tentacles are all considered to be point-like (i.e. they have no size). Their locations are all distinct.
The captain and all pirates move to their target in a straight line at their given speed and are not hindered by anyone or anything.

Output

For every test case in the input, the output should contain one floating point number on a single line: the minimum time it takes for the captain to kill the n-pus. Your answer should have either an absolute or a relative error of at most 10^{-6}.

[1]An n-pus is like an octopus, but then with n tentacles.

Example

Input	Output
3	3.5
3 3	2.802775638
2 0 1	1.5
0 0 2	
1 0 3	
3 0 4	
2 3	
0 1	
1 1	
4 1	
1 3	
0 0 1	
3 0 1	
4 0 1	
7 0 2	
0 1	
4 2	
3 3	
0 0 2	
2 0 3	
3 0 1	
4 0 2	
0 1	
3 1	
4 1	
5 1	

D Polly Wants a Cracker

Problem

In the pirate society, each pirate captain is obligated to obtain a pet. This pet should be at the side (or on the shoulder) of a pirate captain at all times, especially during conversations with good doers who wish to challenge the captain's evil ways.

Greedbeard is the infamous and feared captain of the pirate ship Greatlooter. His pet is the infamous and feared parrot known as Polly. As a parrot, Polly likes to mimic the conversations he overhears. Unfortunately, Polly is not very good at mimicking the sentences he hears. For this reason, Greedbeard has taken it upon himself to assist Polly as he learns to speak the human tongue.

Greedbeard noted that Polly is able to speak whole sentences, but often makes one or more of the three following mistakes:

1. Polly mixes up the sentence by placing words in the wrong order. E.g. instead of saying *"polly wants a cracker"*, Polly will say *"polly cracker a wants"*;
2. Polly forgets some words in a sentence. E.g. instead of saying *"polly wants a cracker"*, Polly will say *"polly a cracker"*;
3. Finally, Polly mixes up, adds or removes consonants and vowels[2] in his words. So instead of saying *"polly wants a cracker"*, Polly will say *"polly wantsu a trackets"*.

Note that Polly never mimics the same word twice in a sentence. The captain always knows the original sentence that Polly is trying to mimic.

Captain Greedbeard is not interested in the first two mistakes. However, mixing up letters or adding/removing letters from a word makes his blood boil. Therefore, each time Polly makes mistakes in a sentence by mixing up letters, the captain will take away one cracker from his lunch, for each letter he mixes up. The number of crackers that the captain removes per word, is the minimum number of edits needed to transform the spoken word into the word that Polly was trying to say. An edit is defined as either an insertion, deletion or substitution of a single letter.

It is not always clear how words in the mimicked sentence correspond to words in the original sentence. Greedbeard assumes the mimicked words are matched with the original words in such a way that the total number of mistakes is minimal.

For instance, if Polly should have said: *"polly wants a cracker"* and says: *"polly crackets wantsu"*, the captain will remove 3 crackers from his lunch. These are 2 crackers for messing up the word *"cracker"* and 1 cracker for adding a letter to the word *"wants"*. He does not care that the word *"a"* is missing, or that words are spoken in the wrong order.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

[2]Read: mixes up, adds or removes letters

- A line containing a single string ($1 \leq$ length ≤ 1000), representing the sentence that Polly is trying to mimic;
- A line containing a single string ($1 \leq$ length ≤ 1000), the sentence that Polly has spoken.

Additional notes:

- The sentence Polly mimics consists of one to eight words.
- A word is a sequence of lowercase letters. Each word ends with a space, or with the end of the sentence.
- A sentence does not contain a sequence of multiple spaces.

Output

For every test case in the input, the output should contain one integer on a single line: the number of crackers that Greedbeard will withhold from Polly's lunch.

Examples

Input	Output
3 polly wants a cracker polly crackets wantsu pretty polly petty polli pieces of eight eight of pieces	3 2 0

E Rolling Dice

Problem

A clan of ninjas have hit upon a strange map: it is divided into squares of size 0.5 by 0.5 inch approximately. Somewhere in the middle, on a spot known to be a foothold of pirates, the square is denoted by 1, while the adjacent squares from north to west bear the numbers 2, 3, 5, and 4, respectively. This strange numbering puzzled the ninjas for a long time until the brightest gambler among them discovered that it had something to do with dice: if the top shows 1, and you roll – or rather tilt – the die to the right onto the neighboring square, then 3 appears on top (depending of course on the orientation of the die.) Together with the map, they laid their hands on a strange text first attributed to the Bay Area Poets Coalition, but it seemed unlikely that even Joyce concocted such a hermetic verse.

The first sentence reads "5X-YYX+Y.". They found out what it means:

- X means one square to the east.
- Y means one square to the north.
- 3X is an abbreviation for XXX, and 3XYX means XXXYX.

The minus sign means that the directions are reversed (east becomes west and north becomes south), the plus sign restores the original directions. The sequence ends with a dot '.'. The number of dots on the top of the die in its final position signifies what is to find on this spot: a hut, a hiding place for weapons, a well, or even a place of unspeakable horror.

Your task is to compute the coordinates of the final position (relative to the starting point) and to determine what number of dots is shown on the top.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- A single line with a string describing a sequence.

A sequence can contain digits, '+', '-', 'X', 'Y', and ends with a dot. A sequence contains at most 1000 characters. The numbers in the sequence consist of at most 7 digits and are never zero.

Output

For every test case in the input, the output should be a single line containing "position (x, y), z dots", where (x, y) is the final position of the die and z the number of dots shown on top of the die in the final position.

Examples

Input	Output
3 5X-YYX+Y. XY-XY. 12X12Y-12X12Y.	position (4,-1), 5 dots position (0,0), 3 dots position (0,0), 1 dots

This page is intentionally left blank.

F Dividing the Loot

Problem

You have successfully led a group of pirates in taking over a commercial vessel. You have seized gold coins, silver coins, and other precious goods. Now it comes down to dividing the loot. It is important to keep everyone satisfied, otherwise you risk mutiny. A pirate is not satisfied if another pirate gets more items than he does. Therefore you might have to content yourself with fewer items than the other pirates or throw some items into open sea. Fortunately, the other pirates have no notion of the value of the items, while you do. Can you make the most out of this without mutiny?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers P and N, satisfying $0 \leq P \leq 1000$, $1 \leq N \leq 1000$: the number of other pirates you have to share the loot with and the number of items, respectively.
- One line with N integers, v_i, satisfying $1 \leq v_i \leq 1000$: the value of each item.

Output

For every test case in the input, the output should contain one integer on a single line: the maximum total value of all the items that you can give yourself, while keeping all other pirates satisfied.

Example

Input	Output

2
2 7
1 1 1 1 3 3 7
5 9
2 2 4 4 6 8 11 11 13
	10
	13

This page is intentionally left blank.

G Secret Island Base

Problem

A group of ninjas want to set up a new secret base for training. For its location, they have selected a remote group of islands. In order to keep the base as covert as possible, they want to build it as far away from the coast as possible. To determine how suitable each island is in this respect, the ninjas have come to you for help.
You must write a program which is given the islands; they are modeled as polygons in flat space. For each island they wish to know the furthest away one could possibly get from the coastline. The distance from a point to the coastline is defined as the shortest Euclidean distance (i.e. distance in a straight line) from this point to a point on the coastline.
Of course, this must remain a secret. They promise they won't kill you, but do not discuss this with anyone! Upsetting a ninja is the last thing you (want to) do in this world.

Input

The first line of the input contains a single number: the number of islands to follow. Each island is given as follows:

- One line with one integer N, satisfying $3 \leq N \leq 20$: the number of vertices of the polygon describing the coastline of the island.
- N lines, each with two integers x and y, satisfying $-100 \leq x, y \leq 100$: the coordinates of each vertex.

The vertices are given in counterclockwise order.

Output

For every island in the input, the output should contain a single floating point number on a single line: the largest distance over which one can be separated from the coastline, while being on the island. Your answer should have either an absolute or a relative error of at most 10^{-3}.

Example

Input	Output

1
3
0 0
10 0
3 8
	2.7421536261

This page is intentionally left blank.

H Stealth Ninja

Problem

A large palace contains a hall, of which the floor plan can be described as a $(2n-1)$ by $(2n-1)$ grid of squares, arranged in $2n-1$ rows numbered from 1 to $2n-1$, and in $2n-1$ columns numbered from 1 to $2n-1$—see the figure below for an example with $n = 4$. Each square that is on the intersection of an even-numbered row and an even-numbered column is covered by a square wooden support column. The other squares are empty. Thus, between the walls and the wooden supports, there are n corridors from left to right and n corridors from back to front through the hall. At the ends of the back-to-front corridors, there are door openings with curtains. Otherwise the walls are closed.

The hall is guarded by k guards. Each of these guards is stationed at a crossing of a front-to-back and a left-to-right corridor. Each of these guards looks 4 seconds in the direction of the left side of the hall, 4 seconds to the back of the hall, 4 seconds to the right, and then 4 seconds to the front, and repeats this cycle indefinitely. However, the guards are not necessarily in phase: for example, some guards may start the cycle by looking to the left, while others start by looking to the front.

A ninja must traverse the hall from front to back without meeting a guard and without being seen. Before entering the hall, the ninja waits in row 0, that is, behind any of the curtains in the door openings of the front wall. He can choose behind which curtain to wait, and for how long, until he enters the hall. The ninja can choose any door in the back wall as the place to exit. The ninja takes 2 seconds to walk from one square to an adjacent square—while walking, the ninja would be seen by any guard that is currently observing one of these squares. Can the ninja traverse the hall successfully? Note that the guards can see past each other, but not through curtains.

Figure 1 on the next page shows an example, in which the ninja finds a way through. First, the ninja waits. After 8 seconds, the guard in row 5, column 1, turns its face to the left, and the ninja steps through the curtain. After 10 seconds, the ninja arrives at row 1, column 1. After 12 seconds, the ninja arrives at row 2, column 1. The ninja waits there until the guard at row 3, column 5 turns its face to the back of the hall; then the ninja starts walking and disappears behind the curtain in column 3 just before the guard in row 1 turns its face in that direction.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integer numbers: n (the number of corridors in each direction, $1 \leq n \leq 250$) and k (the number of guards, $0 \leq k \leq 500$).
- k lines, one for each guard: each row contains an odd integer number r ($1 \leq r \leq 2n-1$), an odd integer number c ($1 \leq c \leq 2n-1$), and one of the letters **L**, **B**, **R**, or **F**. The first number is the row in which the guard is placed; the second number is the column in which the guard is placed, and the letter indicates in which direction the guard is looking at time zero (left, back, right, or front).

There is at most one guard at any particular position in the hall.

Output

For every test case in the input, the output should contain a single line with either the word *succeeds*, or the word *fails*.

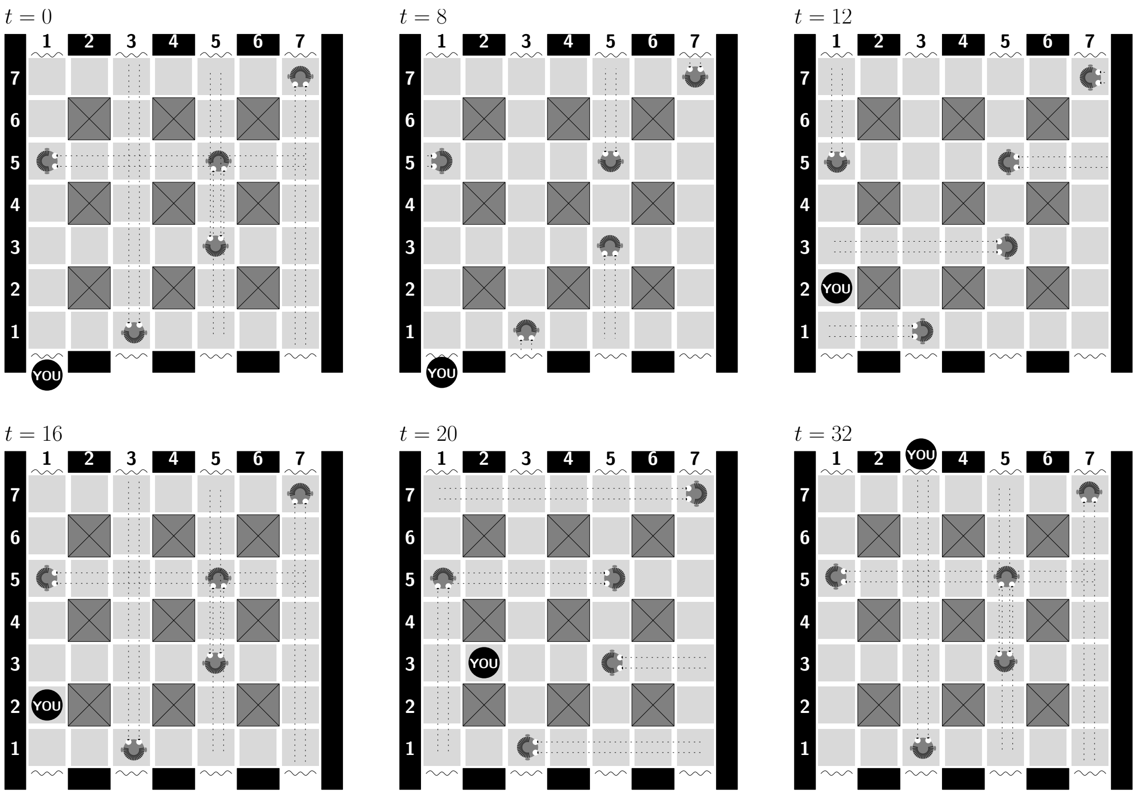

Figure 1: The ninja succeeds in traversing the hall from front to back without being seen.

Example

The following example describes the test case in the figure.

Input	Output
1	succeeds
4 5	
1 3 B	
3 5 B	
5 1 R	
5 5 F	
7 7 F	

I In and Out

Problem

The young pirate Will Twister has lost his precious medallion, once given to him by his father. It is now in the hands of governor Goose. Since this medallion is very dear to Will, he decides to steal it back. His presence will not go unnoticed (he's not a ninja), hence to increase his chances of a successful getaway he will use the cover of night. That does mean though that the journey to the governor's house will also be dangerous, because someone walking through town at night is suspicious.
Throughout the town, sentries[3] are placed at strategically chosen junctions. Will is not a very good fighter, hence he will rely on the element of surprise and his speed to make it past them. However, if he were to pass the same sentry on both the journey to and from the governor's house, the element of surprise would be gone the second time and there is a big risk that he gets caught. Therefore, he does not want to pass the same sentry twice.
Will's journey will start and end at the harbor, where he will arrive and leave by boat. He has a map of the town, complete with the locations of the sentries. Given that he'll have to do a lot of running, he wishes to find the shortest route to and from the governor's house that doesn't take him twice past any sentry. Can you help him determine this route?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers N and R, satisfying $2 \leq N \leq 1000$ and $1 \leq R \leq 10,000$: the number of junctions and the number of roads, respectively.
- R lines, each with three integers a, b and l, satisfying $1 \leq a, b \leq N$ and $1 \leq l \leq 1000$, indicating that there is a bidirectional road between junctions a and b of length l.
- One line with one integer S, satisfying $0 \leq S \leq \min(N-2, 100)$: the number of sentries.
- One line with S distinct integers, satisfying $2 \leq s_i < N$, indicating that there is a sentry at junction s_i.

The junctions are numbered 1 through N. Will's boat is at junction 1 and the governor's house is at junction N. A path from the boat to the governor's house is guaranteed to exist.

Output

For every test case in the input, the output should contain one integer on a single line: the minimum total distance Will has to cover. If there is no route that passes each sentry at most once, the output should be "No safe route" (without the quotation marks) on a single line.

[3]A sentry is a stationary guard

Example

Input	Output
3	42
6 7	8
1 2 1	No safe route
2 3 1	
3 6 1	
1 4 10	
4 3 10	
2 5 10	
5 6 10	
2	
2 3	
5 5	
1 2 1	
1 3 2	
2 4 1	
3 4 2	
4 5 1	
1	
2	
5 5	
1 2 1	
1 3 2	
2 4 1	
3 4 2	
4 5 1	
1	
4	

J Shuriken Game

Problem

Being a ninja means spending a lot of time training. To pass the time between training sessions, ninjas like to play games with their shurikens[4]. An old favorite of the ninjas is a game for two players featuring one stack of shurikens. In turn, each player takes a number of shurikens from the stack, at least one but at most N. The winner is the player to take the last shuriken(s). It took a while, but every ninja eventually discovered what strategy you have to follow in order to win, and the ninjas lost interest in the game.

Fortunately, one ninja came up with an additional rule that would hopefully make the game non-trivial again. This extra rule is that a player is not allowed to copy his opponent's last move, i.e. remove the same number of shurikens as his opponent just did. If the stack consists of just one shuriken, and the other player has just taken one, the player to move has no legal move and loses. Can you help the ninjas figure out how the player to move can win in a given game situation?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with three integers S, N and P, satisfying $1 \leq S \leq 100,000$, $2 \leq N \leq 100$ and $1 \leq P \leq N$: the number of shurikens in the stack, the maximum number of shurikens a player is allowed to take from the stack, and the last move from the other player, respectively.

Output

For every test case in the input, the output should contain one integer on a single line: the smallest amount of shurikens that the player to move can take in order to secure the win. If there is no winning move, the output should be 0.

Example

Input	Output
5	2
12 4 1	0
5 5 5	3
6 6 6	1
100 5 5	2
100 5 1	

[4] A shuriken is a metal star with sharp corners which ninjas use to throw at enemies

This page is intentionally left blank.