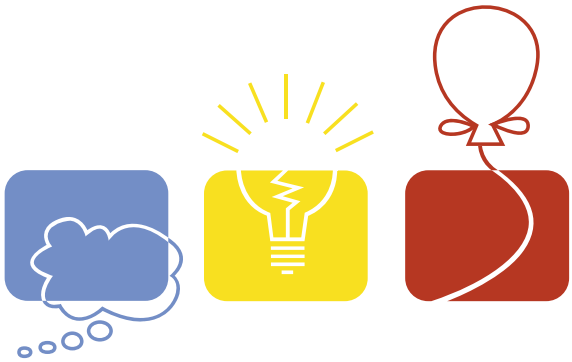
NWERC 2004

The 2004 ACM Northwestern Europe Programming Contest
Lund Institute of Technology, Lund, Sweden



acm International Collegiate Programming Contest

IBM event sponsor

The Problem Set

A Word Encoding
B Watchdog
C Taxi Cab Scheme
D Pseudo-random Numbers
E Card Game Cheater
F Investment
G Pipes
H SETI
I Minimax Triangulation

Blank page

Problem A: Word Encoding

In any language, certain combinations of letters do not appear (or at least appear so seldom that they can be considered non-existent). For instance, there are no English words containing the three letter combination buv as a substring. Given a list of letter combinations that do not exist, the number of possible "words" in a language can be reduced a lot (a "word" here means any combination of letters that doesn't contain any of the given letter combinations as a substring). If we order all such words by increasing length, ordering words of the same length alphabetically, we can enumerate them starting from 1. Assume that the alphabet always consists of the lower case letters 'a' to 'z'.

For instance, if the list only contains the combinations q, ab and aaa, the words would be enumerated like this:

1. a
2. b

⋮

16. p
17. r

⋮

26. aa
27. ac

⋮

649. zz
650. aac

Given the list of letter combinations, write a program that for a given word outputs its number, and for a given number ouputs its word. You can assume that none of the words will exceed 20 characters and no number will be greater than 2 000 000 000 (for both input and output).

Input

The input will contain several test cases. The number of test cases T appears on a line by itself. Then follow T test cases. Each test case starts with a line containing two integers, N (the number of letter combinations, non-negative, at most 1 000) and M (the number of queries for this list, positive, at most 100). Then follow N lines, each containing a lower case letter combination (between 1 and 3 letters, inclusive). After that follow M lines, each containing either a positive integer or a lower case word. If it's a word, it will not contain any of the combinations of letters in the list for this test case. If it's a number, it will not be greater than the number of words in the language.

Output

For each query, output a single line containing either the word's corresponding number, or the number's corresponding word.

Sample input	**Sample output**
2 3 4 q ab aaa 16 r 27 aac 7 2 a b c d ef ghi ijk 102345678 ksvfuw	p 17 ac 650 xexgun 39174383

Problem B: Watchdog

A company (name withheld) has an office building in the center of Lund. The building has a perfectly square roof with a number of hatches. Because of a series of burglaries where the perpetrators have entered through these hatches, it was decided to use a watchdog to guard the hatches. A particularly vicious but rather stupid breed of dog was chosen, and unfortunately the dog fell off the roof on its third watch.

A new dog has been procured and it has been decided to attach a leash to its collar and attach the other end at some point on the roof. However, if the leash is too short the dog cannot reach all hatches, but if it is too long then the dog will fall off the building again. The leash has hooks at both ends, so no part of it is used to tie knots. The company wants the dog to reach the center of each hatch (the dog can reach exactly as far as the leash could reach if it were lying flat on the roof), but it does not want the leash to extend beyond the edge of the roof (to the edge is OK). They hope that by carefully choosing the length of the leash and where to attach it, the dog will be able to reach all hatches without risking falling off the building. A leash can only be attached at a point with integer coordinates (if the building is 10 by 10 meters, then the south-west corner of the building has coordinates $(0, 0)$ and the north-east corner has coordinates $(10, 10)$). A leash cannot be attached at a point where there is a hatch.

If there is no place where you can attach a leash, reach all hatches but not reach beyond the edge of the roof, it is impossible to use this breed of dog, and the company will instead use a poodle (which is a less vicious type of dog, but also less prone to falling off buildings).

Input

On the first line of the input is a single positive integer N, telling the number of test cases to follow. Each case starts with one line with two integers S H, where S is even, $2 \leq S \leq 40$, and $1 \leq H \leq 50$. S is the side of the square roof in meters and H is the number of hatches. The following H lines each contain two integers X and Y. These are the coordinates of the hatches. Hatches will never lie outside the roof or on the roof's perimeter. No two hatches will occupy the same position

Output

For each test case, output one line containing the coordinates X Y at which to fasten the leash (if there are several possible points, output the one with smallest X, and if there are still several possibilities select the one with smallest Y among those with smallest X) such that a leash of suitable length allows access to all hatches without extending beyond the edge of the roof. If there is no such point, output "poodle" for that test case.

Sample input	**Sample output**
3 10 2 6 6 5 4 20 2 1 1 19 19 10 3 1 1 1 2 1 3	3 6 poodle 2 2

Problem C: Taxi Cab Scheme

Running a taxi station is not all that simple. Apart from the obvious demand for a centralised coordination of the cabs in order to pick up the customers calling to get a cab as soon as possible, there is also a need to schedule all the taxi rides which have been booked in advance. Given a list of all booked taxi rides for the next day, you want to minimise the number of cabs needed to carry out all of the rides.

For the sake of simplicity, we model a city as a rectangular grid. An address in the city is denoted by two integers: the street and avenue number. The time needed to get from the address a, b to c, d by taxi is $|a - c| + |b - d|$ minutes. A cab may carry out a booked ride if it is its first ride of the day, or if it can get to the source address of the new ride from its latest, at least one minute before the new ride's scheduled departure. Note that some rides may end after midnight.

Input

On the first line of the input is a single positive integer N, telling the number of test scenarios to follow. Each scenario begins with a line containing an integer M, $0 < M < 500$, being the number of booked taxi rides. The following M lines contain the rides. Each ride is described by a departure time on the format `hh:mm` (ranging from 00:00 to 23:59), two integers *a b* that are the coordinates of the source address and two integers *c d* that are the coordinates of the destination address. All coordinates are at least 0 and strictly smaller than 200. The booked rides in each scenario are sorted in order of increasing departure time.

Output

For each scenario, output one line containing the minimum number of cabs required to carry out all the booked taxi rides.

Sample input	**Sample output**
2 2 08:00 10 11 9 16 08:07 9 16 10 11 2 08:00 10 11 9 16 08:06 9 16 10 11	1 2

Blank page

Problem D: Pseudo-random Numbers

Access to high-quality randomness is very important for many applications, especially in cryptography. Radioactive decay is somtimes used as a source of "true randomness", but this is a fairly slow procedure for getting random numbers. Also, in many applications it is important that the same "random" sequence can be produced in two different places. For these reasons one often uses a *pseudo-random* sequence instead. A pseudo-random sequence is a sequence that is, in fact, not random, but very hard to distinguish from a truly random sequence. A pseudo-random sequence should also be difficult to predict, i.e., given the first few elements of the sequence it should be difficult do determine some later, yet unseen, number in the sequence.

The Association of Cryptographic Machinery (ACM) has devised an algorithm for generating pseudo-random number sequences, but they have no idea how good it really is. Therefore they want you to test it.

The algorithm to generate a sequence of integers, where each integer is between 0 and $B-1$ inclusive, is as follows:

1. Start with any number (the seed) in base B. This number can contain hundreds of base B digits.

2. The last digit (least significant) is output as the next element of the sequence.

3. Create a new number by writing down the sum of all neighbouring digits from left to right. E.g., with $B = 10$, the number 845 would yield the number 129 (since $8+4=12$ and $4+5=9$).

4. Repeat steps 2 and 3 as many times as needed, or until the number has only one base B digit. You get one more pseudo-random digit between 0 and $B-1$ each time.

If we have $B = 10$ and the seed number is 845, then the next numbers will be 129, 311 ($1+2=3$, $2+9=11$), 42 ($3+1=4$, $1+1=2$), and 6 ($4+2=6$). As 6 is a single digit base 10 number, the algorithm terminates. The pseudo-random digits generated are 5, 9, 1, 2 and 6.

You will be testing the generator as follows. You will be given the first L elements output by the generator and an integer $T > L$. You are supposed to decide if the first T elements are completely determined by the first L elements. To check the robustness of your testing procedure the ACM have slipped in some *impossible* sequences, i.e. sequences that cannot be generated by any initial seed.

Input

On the first line of the input is a single positive integer N, telling the number of test cases to follow. The first line of each test case consists of one integer B ($2 \leq B \leq 1\,000$), the base. The second line consists of an integer L ($1 \leq L \leq 100$), followed by the L first elements of some sequence (the elements are written in base 10 and are between 0 and $B-1$ inclusive). The third line consists of an integer T, ($L < T \leq 100\,000$), the element of the sequence to predict.

Output

For each test case output, on a line of its own:

- “impossible” if no seed number can produce the given sequence.
- “unpredictable” if there exists a seed number that produces the given sequence but the first T elements are not completely determined by the first L elements.
- the T:th element of the sequence in base 10, otherwise.

Sample input	**Sample output**
3 10 5 5 9 6 7 0 7 16 4 11 7 8 4 12 2 5 0 1 1 1 0 10	8 unpredictable impossible

Problem E: Card Game Cheater

Adam and Eve play a card game using a regular deck of 52 cards. The rules are simple. The players sit on opposite sides of a table, facing each other. Each player gets k cards from the deck and, after looking at them, places the cards face down in a row on the table. Adam's cards are numbered from 1 to k from his left, and Eve's cards are numbered 1 to k from her right (so Eve's i:th card is opposite Adam's i:th card). The cards are turned face up, and points are awarded as follows (for each $i \in \{1, \ldots, k\}$):

- If Adam's i:th card beats Eve's i:th card, then Adam gets one point.
- If Eve's i:th card beats Adam's i:th card, then Eve gets one point.
- A card with higher value always beats a card with a lower value: a three beats a two, a four beats a three and a two, etc. An ace beats every card except (possibly) another ace.
- If the two i:th cards have the same value, then the suit determines who wins: hearts beats all other suits, spades beats all suits except hearts, diamond beats only clubs, and clubs does not beat any suit.

For example, the ten of spades beats the ten of diamonds but not the Jack of clubs.

This ought to be a game of chance, but lately Eve is winning most of the time, and the reason is that she has started to use marked cards. In other words, she knows which cards Adam has on the table before he turns them face up. Using this information she orders her own cards so that she gets as many points as possible.

Your task is to, given Adam's and Eve's cards, determine how many points Eve will get if she plays optimally.

Input

There will be several test cases. The first line of input will contain a single positive integer N giving the number of test cases. After that line follow the test cases.

Each test case starts with a line with a single positive integer $k \leq 26$ which is the number of cards each player gets. The next line describes the k cards Adam has placed on the table, left to right. The next line describes the k cards Eve has (but she has not yet placed them on the table). A card is described by two characters, the first one being its value (2, 3, 4, 5, 6, 7, 8 ,9, T, J, Q, K, or A), and the second one being its suit (C, D, S, or H). Cards are separated by white spaces. So if Adam's cards are the ten of clubs, the two of hearts, and the Jack of diamonds, that could be described by the line

```
TC 2H JD
```

Output

For each test case output a single line with the number of points Eve gets if she picks the optimal way to arrange her cards on the table.

Sample input	**Sample output**
3 1 JD JH 2 5D TC 4C 5H 3 2H 3H 4H 2D 3D 4D	1 1 2

Problem F: Investment

John never knew he had a grand-uncle, until he received the notary's letter. He learned that his late grand-uncle had gathered a lot of money, somewhere in South-America, and that John was the only inheritor.

John did not need that much money for the moment. But he realized that it would be a good idea to store this capital in a safe place, and have it grow until he decided to retire. The bank convinced him that a certain kind of bond was interesting for him.

This kind of bond has a fixed value, and gives a fixed amount of yearly interest, payed to the owner at the end of each year. The bond has no fixed term. Bonds are available in different sizes. The larger ones usually give a better interest. Soon John realized that the optimal set of bonds to buy was not trivial to figure out. Moreover, after a few years his capital would have grown, and the schedule had to be re-evaluated.

Assume the following bonds are available:

Value	Annual interest
4000	400
3000	250

With a capital of €10 000 one could buy two bonds of €4 000, giving a yearly interest of €800. Buying two bonds of €3 000, and one of €4 000 is a better idea, as it gives a yearly interest of €900. After two years the capital has grown to €11 800, and it makes sense to sell a €3 000 one and buy a €4 000 one, so the annual interest grows to €1 050. This is where this story grows unlikely: the bank does not charge for buying and selling bonds. Next year the total sum is €12 850, which allows for three times €4 000, giving a yearly interest of €1 200.

Here is your problem: given an amount to begin with, a number of years, and a set of bonds with their values and interests, find out how big the amount may grow in the given period, using the best schedule for buying and selling bonds.

Input

The first line contains a single positive integer N which is the number of test cases. The test cases follow.

The first line of a test case contains two positive integers: the amount to start with (at most €1 000 000), and the number of years the capital may grow (at most 40).

The following line contains a single number: the number d ($1 \leq d \leq 10$) of available bonds.

The next d lines each contain the description of a bond. The description of a bond consists of two positive integers: the value of the bond, and the yearly interest for that bond. The value of a bond is always a multiple of €1 000. The interest of a bond is never more than 10% of its value.

Output

For each test case, output – on a separate line – the capital at the end of the period, after an optimal schedule of buying and selling.

Sample input	**Sample output**
1 10000 4 2 4000 400 3000 250	14050

Problem G: Pipes

The construction of office buildings has become a very standardized task. Pre-fabricated modules are combined according to the customer's needs, shipped from a faraway factory, and assembled on the construction site. However, there are still some tasks that require careful planning, one example being the routing of pipes for the heating system.

A modern office building is made up of square modules, one on each floor being a service module from which (among other things) hot water is pumped out to the other modules through the heating pipes. Each module (including the service module) will have heating pipes connecting it to exactly two of its two to four neighboring modules. Thus, the pipes have to run in a circuit, from the service module, visiting each module exactly once, before finally returning to the service module. Due to different properties of the modules, having pipes connecting a pair of adjacent modules comes at different costs. For example, some modules are separated by thick walls, increasing the cost of laying pipes. Your task is to, given a description of a floor of an office building, decide the cheapest way to route the heating pipes.

Input

The first line of input contains a single integer, stating the number of floors to handle. Then follow n floor descriptions, each beginning on a new line with two integers, $2 \leq r \leq 10$ and $2 \leq c \leq 10$, defining the size of the floor – r-by-c modules. Beginning on the next line follows a floor description in ASCII format, in total $2r + 1$ rows, each with $2c + 2$ characters, including the final newline. All floors are perfectly rectangular, and will always have an even number of modules. All interior walls are represented by numeric characters, '0' to '9', indicating the cost of routing pipes through the wall (see sample input).

Output

For each test case, output a single line with the cost of the cheapest route.

Sample input	**Sample output**
3 4 3 ####### # 2 3 # #1#9#1# # 2 3 # #1#7#1# # 5 3 # #1#9#1# # 2 3 # ####### 4 4 ######### # 2 3 3 # #1#9#1#4# # 2 3 6 # #1#7#1#5# # 5 3 1 # #1#9#1#7# # 2 3 0 # ######### 2 2 ##### # 1 # #2#3# # 4 # #####	28 45 10

Problem H: SETI

For some years, quite a lot of work has been put into listening to electromagnetic radio signals received from space, in order to understand what civilizations in distant galaxies might be trying to tell us. One signal source that has been of particular interest to the scientists at *Université de Technologie Spatiale* is the Nebula Stupidicus.

Recently, it was discovered that if each message is assumed to be transmitted as a sequence of integers $a_0, a_1, ...a_{n-1}$ the function $f(k) = \sum_{i=0}^{n-1} a_i k^i \pmod{p}$ always evaluates to values $0 \leq f(k) \leq 26$ for $1 \leq k \leq n$, provided that the correct value of p is used. n is of course the length of the transmitted message, and the a_i denote integers such that $0 \leq a_i < p$. p is a prime number that is guaranteed to be larger than n as well as larger than 26. It is, however, known to never exceed 30 000.

These relationships altogether have been considered too peculiar for being pure coincidences, which calls for further investigation.

The linguists at the faculty of *Langues et Cultures Extraterrestres* transcribe these messages to strings in the English alphabet to make the messages easier to handle while trying to interpret their meanings. The transcription procedure simply assigns the letters *a*..*z* to the values 1..26 that $f(k)$ might evaluate to, such that $1 = a$, $2 = b$ etc. The value 0 is transcribed to '*' (an asterisk). While transcribing messages, the linguists simply loop from $k = 1$ to n, and append the character corresponding to the value of $f(k)$ at the end of the string.

The backward transcription procedure, has however, turned out to be too complex for the linguists to handle by themselves. You are therefore assigned the task of writing a program that converts a set of strings to their corresponding Extra Terrestial number sequences.

Input

On the first line of the input there is a single positive integer N, telling the number of test cases to follow. Each case consists of one line containing the value of p to use during the transcription of the string, followed by the actual string to be transcribed. The only allowed characters in the string are the lower case letters 'a'..'z' and '*' (asterisk). No string will be longer than 70 characters.

Output

For each transcribed string, output a line with the corresponding list of integers, separated by space, with each integer given in the order of ascending values of i.

Sample input	**Sample output**
3 31 aaa 37 abc 29 hello*earth	1 0 0 0 1 0 8 13 9 13 4 27 18 10 12 24 15

Blank page

Problem I: Minimax Triangulation

Triangulation of surfaces has applications in the Finite Element Method of solid mechanics. The objective is to estimate the stress and strain on complex objects by partitioning them into small simple objects which are considered incompressible. It is convenient to approximate a plane surface with a *simple polygon*, i.e., a piecewise-linear, closed curve in the plane on m distinct vertices, which does not intersect itself. A *chord* is a line segment between two non-adjacent vertices of the polygon which lies entirely inside the polygon, so in particular, the endpoints of the chord are the only points of the chord that touch the boundary of the polygon. A *triangulation* of the polygon, is any choice of $m-3$ chords, such that the polygon is divided into triangles. In a triangulation, no two of the chosen chords intersect each other, except at endpoints, and all of the remaining (unchosen) chords cross at least one of the chosen chords. Fortunately, finding an arbitrary triangulation is a fairly easy task, but what if you were asked to find the best triangulation according to some measure?

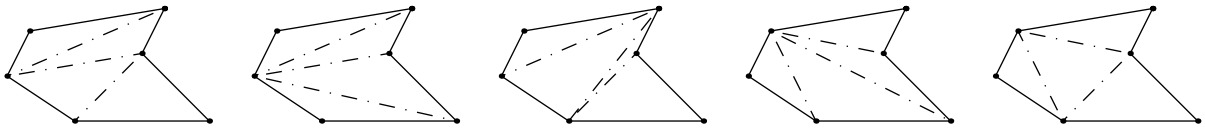


Figure I.1: Five out of nine possible triangulations of the example polygon. The leftmost has the smallest largest triangle.

Input

On the first line of the input is a single positive integer n, telling the number of test scenarios to follow. Each scenario begins with a line containing one positive integer $2 < m < 50$, being the number of vertices of the simple polygon. The following m lines contain the vertices of the polygon in the order they appear along the border, going either clockwise or counter clockwise, starting at an arbitrary vertex. Each vertex is described by a pair of integers $x\ y$ obeying $0 \leq x \leq 10\,000$ and $0 \leq y \leq 10\,000$.

Output

For each scenario, output one line containing the area of the largest triangle in the triangulation of the polygon which has the smallest largest triangle. The area should be presented with one fractional decimal digit.

Sample input	**Sample output**
1 6 7 0 6 2 9 5 3 5 0 3 1 1	9.0