MERCER UNIVERSITY

Spring Programming Contest

February 24, 2018

Arena Survival	(prob1)
Best Burger in Macon	(prob2)
Clump Finding	(prob3)
Comedy Show	(prob4)
Dog Park	(prob5)
Drizzle	(prob6)
Great Race	(prob7)
Kangaroo Words	(prob8)
Problem 9 was removed from the contest.	(prob9)
Nth Derivative	(prob10)
Schrődinger’s Character	(prob11)
Unfinished Crossword	(prob12)

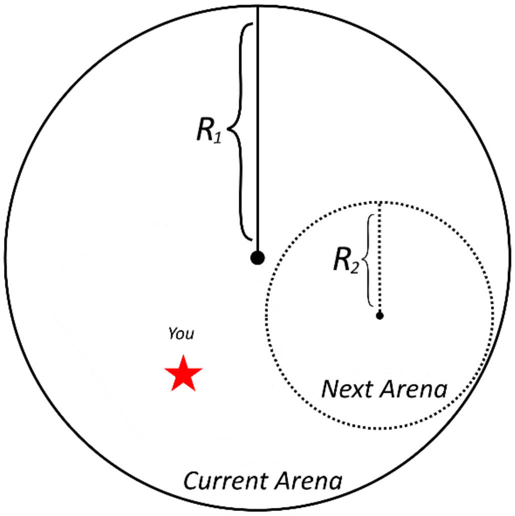
The name in parenthesis following each problem is the name you must use as your program’s name. You must add the appropriate extension depending upon your choice of programming language (.c .cpp .cs .java .py).

Arena Survival (prob1)

The Problem

You are playing a very popular video game where you are placed in a circular arena with many other competitors. The goal is to be the last one standing while the arena slowly shrinks in size, bringing combatants closer together over time. You've noticed that the way the arena shrinks follows a very simple pattern. Every 5 minutes, the current circular arena shrinks to a smaller circle that is completely contained within the current arena. You figured out the current arena's radius R_1 and the next arena's radius R_2. Unfortunately, you don't know where the next circle is going to be until the current arena starts shrinking. You do know that it has a uniformly distributed probability of being any spot that would result in the next arena being fully contained in the current arena.

Afraid that your current spot is not likely to be in the next circle, you have decided to write a program that will take your current position and return the probability that you will be contained in the next circle.



The Input

The first line will contain an integer *T* for the number of test cases. The following *T* lines will contain 4 real numbers R_1, R_2, *X*, and *Y* ($1 \le R_2 \le R_1 \le 1000$ and $-1000 \le X, Y \le 1000$) representing the radius of the current arena, the radius of the next arena, and the position you are currently in, respectively. Your position is guaranteed to be contained within the current arena which is centered around the point (0,0). The border of a circle is considered contained within that circle.

The Output

For each input case, output a single decimal number representing the probability that you will be contained in the next arena rounded to 2 decimal places.

Sample Input

```
3
2 1 0 0
5 2 1 1
5.3 1.4 1.55 3.1
```

Sample Output

```
1.00
0.41
0.08
```

Best Burger in Macon (prob2)

The Problem

You are in Macon for the Mercer Programming Contest and love hamburgers. In order to best prepare yourself for the contest, you'd like to go to the best burger joint in Macon! Unfortunately, not all of the critics agree on which place is the best. Naturally, each place advertises their best rating. For example, if one critic rated Locos as the fourth best burger place in Macon and another rated it as the second best in Macon, Locos would simply claim they were the second best in Macon. In this problem, each of *n* burger joints are ranked by each of *k* critics from first through worst (nth rank). For each burger joint, you are to determine its best rating and the critic who gave it. If multiple critics gave a burger joint its highest rank, select the lowest numbered critic who gave that rank, since the critics are given in order of importance.

Input

The first line of input will contain a single positive integer, *c* ($c \le 30$), representing the number of input cases to process. The input cases follow. The first line of each input case has two positive integers, *n* ($n \le 100$), representing the number of burger joints in town and *k* ($k \le 20$), representing the number of food critics ranking the burger joints. The burger joints are numbered 1 to *n* and the food critics are numbered 1 to *k*. The following *k* lines contain the rankings of the *k* critics. The ith ($1 \le i \le k$) of these lines contains the ranking information for all burger joints by the ith critic. Each of these lines will contain a space separated permutation of the integers 1 through *n*, inclusive, which represent the burger joints ordered by that critic's ranks. Namely, the first integer on the ith of these lines represents the 1st ranked burger joint by the ith critic, the second of these integers represent the 2nd ranked burger joint by the ith critic and so forth.

Output

For each input case, output *n* lines, one per burger joint, in order of the number of the burger joint. For each burger joint, output two space separated integers: *r*, their best rank achieved, and *c*, the lowest numbered critic who gave that burger joint that ranking. (Note: best rank means the lowest numbered rank.)

Sample Input	**Sample Output**
`2`	`2 1`
`4 3`	`1 1`
`2 1 4 3`	`1 3`
`2 4 1 3`	`2 2`
`3 2 4 1`	`2 4`
`6 4`	`1 1`
`2 3 1 6 5 4`	`1 2`
`3 2 6 1 4 5`	`5 2`
`2 3 1 6 5 4`	`5 1`
`6 1 2 3 4 5`	`1 4`

Clump Finding (prob3)

The Problem

A genome is represented by a string of the alphabet (A, C, G, T), and an example is Vibrio cholerae, the bacterium that causes cholera;

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctattttttacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

A string of length k, called *k-mer*, that appears multiple times in a small window of a genome could mean something useful, such as replication region in the genome. We defined a k-mer as a “clump” if it appears many times within a short interval of the genome. More formally, given integers *L* and *t*, a k-mer Pattern forms an *(L, t)-clump* inside a (larger) string Genome if there is an interval of Genome of length *L* in which this k-mer appears at least *t* times. For example, TGCA forms a (25, 3)-clump in the following Genome:

```
gatcagcataagggtcccTGCAaTGCAtgacaagccTGCAgttgttttac
```

From the example of Vibrio cholera, 9-mer (500, 3)-clumps include `atgatcaag`, `ctcttgatc`,`cttgatcat`, and `tcttgatca`.

Input

The first line of input contains a single integer, c (1 ≤ c ≤ 100), the number of test sets.
Each test case will consist of the following, each on a separate line:

- A string Genome,
- Integer, k (2 ≤ k ≤ 20), the length of a clump,
- Integer, L (10 ≤ L ≤ 500), the length of the window where a clump can be formed,
- Integer, t (2 ≤ t), the minimum frequency of the k-mer to form a clump

The input for each test case is separated with a blank line.

Output

For each set, output all distinct k-mers forming (L, t)-clumps in Genome, separated by a space, in alphabetical order.

Separate the output for each test case with a blank line.

Sample Input

```
2
gatcagcataagggtccctgcaatgcatgacaagcctgcagttgttttac
4
25
3
cgacgatttacctcttgatcatcgatccgattgaagatcttcaattgttaattctcttgcctcgactca
tagccatgatgagctcttgatcatgtttccttaaccctctatttttacggaagaatgatcaagctgct
gctcttgatcatcgtttc
9
150
3
```

Sample Output

```
tgca
ctcttgatc cttgatcat tcttgatca
```

Comedy Show (prob4)

The Problem

A popular television network has recently planned a comedy show where several comedians do their stand up routines, in sequence. Each stand up routine has a funny score. To optimize viewership, it's best that any subsequent stand up routine is better than the previous one, to keep viewers wanting more. We define a valid funny score sequence of a particular show to be a sequence $s_1, s_2, ..., s_k$ where s_i $(1 \le i \le k)$ is the funny score of the i^{th} comedian performing for the show, with $0 < s_1 < s_2 < ... < s_k$. Two funny score sequences of length k, $(s_{1,} s_2, s_3, ..., s_k)$ and $(t_{1,} t_2, t_3, ..., t_k)$ are different if there is some value of i, $1 \le i \le k$, with $s_i \ne t_i$.

We define the total funny score of a show to simply be the sum of the funny scores of each comedian on the show. The producers would like the total funny score of the show to be within a lower and upper bound, since if the show isn't funny at all, it'll be canceled, and if it's too funny, its viewership will exceed that of the network flagship show about funny pet tricks.

The producers of the show would like you to make some calculations for them. Given various values of k, the number of comedians per show, the lower bound for the total funny score, L, and the upper bound for the total funny score, U, they would like you to calculate the total number of different possible valid funny score sequences that satisfy the criteria.

Input

The first line of input contains a single positive integer, N ($N \le 50$), the number of show scenarios the producers would like you to consider. The show scenarios follow, one per line, on the following lines. Each show scenario contains three space separated positive integers, k ($k \le 25$), the number of comedians for the show, L ($L \le 1000$), the lower bound for the total funny score, and U ($L \le U \le 1000$), the upper bound for the total funny score.

Output

For each show scenario, output the number of different funny score sequences possible modulo $10^9 + 7$.

Sample Input	Sample Output

2
3 6 8
5 4 20
	4
	19

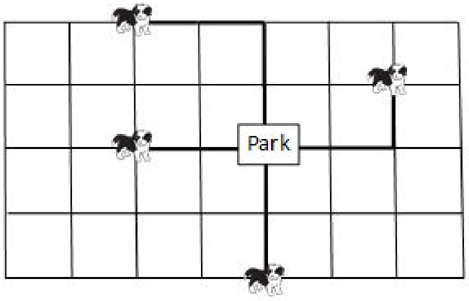
Note: For the first sample input case, the 4 possible sequences are (1, 2, 3), (1, 2, 4), (1, 2, 5), and (1, 3, 4).

Dog Park (prob5)

The Problem

To celebrate the Chinese New Year (the year of the dog), the city council decided to build a new dog park. The dog park is to be located at the point (*x*, *y*) so that the total walking distance of all the dogs from their homes to the dog park is minimized. Because of the layout of the city, the Manhattan distance (*L*^{1} distance) is used to measure the walking distance. If a dog's home is at (*xi*, *yi*), the distance to the dog park is:

$|xi - x| + |yi - y|$



Input

The input consists of a number of test cases. Each case is described by one input line of 2n+1 integers delimited by a space.

```
n x1 y1 x2 y2 … xn yn
```

The first integer *n* represents the number of dogs (0<*n*<=100). It is followed by a list of 2*n* integers representing the coordinates of the dogs (*xi*, *yi*) (0<=*xi, yi* <100000). The input is terminated by a line consisting of the number 0.

Output

For each of the test cases, print an integer for the minimum total Manhattan distance of all the dogs from home to the dog park.

Sample Input

```
3 1 1 2 2 3 3
1 4 3
4 2 2 2 4 4 0 6 3
0
```

Sample Output

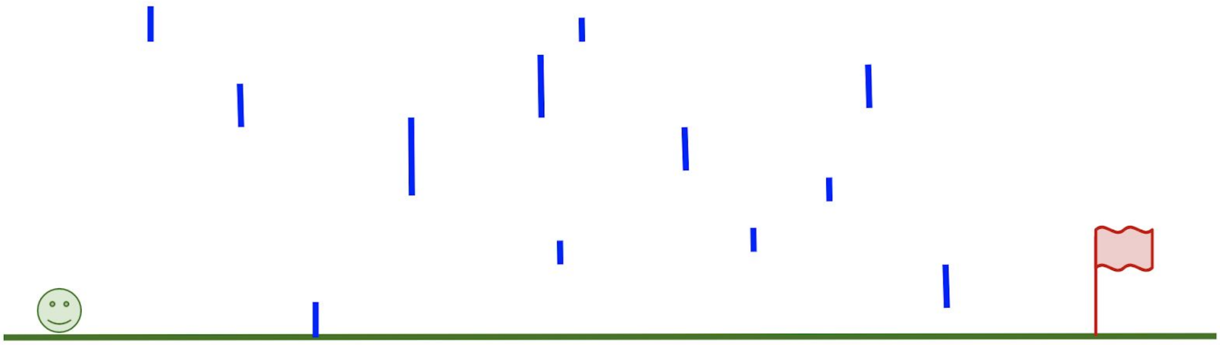
```
4
0
11
```

Drizzle (prob6)

The Problem

“Rain is good for vegetables, and for the animals who eat those vegetables, and for the animals who eat those animals.”

But people sometimes don’t like to get wet. Bally needs to go from home at point 0 to work, which is located in point X, but it’s raining outside. Rain drops in Bally’s world are line segments of different length L_i, that fall with constant speed V_i as the gravity doesn’t exist when it’s raining. Bally can roll with any speed from 0 to V_{MAX} distance units per second, but the speed needs to be constant throughout its journey. What is the earliest time when Bally can make to work without getting wet? Bally is really small and can be considered a point on a plane, and it will get wet, if its coordinates lay ***strictly*** inside the raindrop (ignoring the ends).



Input

The input starts with a single integer T ($T \leq 10000$), the number of test cases.

Each test case starts with integers N ($1 \leq N \leq 1000$), the number of raindrops, X ($0 \leq X \leq 1,000,000$) the position of the work and V_{MAX} ($1 \leq V_{MAX} \leq 1,000,000$ units/sec), the maximum speed (NOTE: Bally can choose to travel with any speed from 0 to V_{MAX}, not necessarily integer).

The i^th^ of the next N lines contain X_i($0<X_i<X$) , Y_i ($0 \leq Y_i \leq 1,000,000$), L_i ($0<L_i \leq 100$) and V_i ($0<V_i \leq 1,000,000$), the current x-coordinate of the bottom of the i^th^ raindrop, the current height of the bottom of the i^th^ raindrop, the length of the i^th^ raindrop and the constant velocity of the i^th^ raindrop, respectively.

Output

For each test case, output the minimum time when Bally can arrive at work. Please output the time with exactly 6 decimal points.

Sample Input

```
3
1 10 10
5 10  10 10
2 30 30
10 0 5 5
20 5 5 5
3 100 10
10 1 13 3
50 25 5 7
75 10 20 3
```

Sample Output

```
1.000000
3.000000
46.666667
```

The Great Race (prob7)

The Problem

Our contenders begin a very long cross country race at the same time. The first contender to reach or pass the finish line is deemed the winner. The course weaves its way up the side of a slippery mountains, so occasionally the contenders slip and lose ground, or stop and take a break. It is possible to slip backwards past the start line.

Animal	Move type	Percent of the time	Actual move
Tortoise			
	Slow plod	50%	1 step forward
	Slip	20%	2 steps backward
	Fast plod	30%	3 steps forward
Hare			
	Sleep	20%	No movement
	Big hop	20%	9 steps forward
	Big slip	10%	12 steps backward
	Small hop	30%	1 step forward
	Small slip	20%	2 steps backward

You will determine the most likely winner of the race based on their average movement speed.

In the above example, the Tortoise averages 1 space forward at a time, and the Hare averages 0.5 spaces forward at a time. So, our expected winner is the Tortoise.

Input

The input consists of multiple races with different animals and different sets of moves.

The first line of each race will begin with the word "Race" followed by a single positive integer *r*, representing the race number being described.

Each race will have two competitors. Each competitor section will begin with the name of the competitor, followed by a series of lines containing each move for the competitor. Each competitor is guaranteed to have at least one move, the name of each competitor will contain only letters and have a length in between 1 and 10, inclusive, and none of the competitors will be named "Race".

Each move line will contain two integers, *p* ($0 < p \leq 100$) and *m* ($|m| \leq$ 10^9) respectively, representing the percentage of the time the competitor does the corresponding move, followed by the number of spaces that move results in. It is guaranteed that the sum of *p* for a competitors set of moves will not exceed 100. If this sum is less than 100, it can be assumed that the rest of the time the competitor does not move.

Input will end when you reach the text “END RACES”

Output

For each race, print the race number and the expected result of who wins, tie, or no winner. A tie is defined as difference in the average speed of two racers being ≤ 0.001.

Sample Input

```
Race 1
Tortoise
50 1
20 -2
30 3
Hare
20 0
20 9
10 -12
30 1
20 -2
Race 2
Hare
20 -6
10 15
Rabbit
20 -3
30 3
Race 3
Iguana
5 -1
Snake
10 -1
END RACES
```

Sample Output

```
Race 1 expected result is Tortoise wins!
Race 2 expected result is a tie!
Race 3 expected result is no winner!
```

Kangaroo Words (prob8)

Write a program that compares two strings, *string1* and *string2*, and determines whether *string1* is a kangaroo word for *string2*. A kangaroo word is a word that contains the letters of another word in order. For example, the string “MASCULINE” is a kangaroo word for the string “MALE” since “MALE” is contained within “MASCULINE” in the correct order.

Input

The input will start with a single line containing a positive integer *n* representing the specified number of pairs of lines to be input. Each pair of lines consists of two strings starting with a single character that is not a white space character (tab, blank, newline). This single character is always a letter, and will be followed by zero or more letters. The last character on each line is a newline character. All strings should be converted to uppercase upon input, and before comparison. Each string will consist of 1 to 80 letters.

Output

Format output as shown below where *string1* and *string2* both appear in uppercase with the string “ `is a kangaroo word for` ” or “ `is not a kangaroo word for` ” are between them. Note that *string1* is always the first string of the two strings to be input.

Sample Input

```
6
football
ball
ALE
Apple
Apple
ALE
Clemson
Son
Allocate
Allot
Mercer
ceer
```

Sample Output

```
FOOTBALL is a kangaroo word for BALL
ALE is not a kangaroo word for APPLE
APPLE is a kangaroo word for ALE
CLEMSON is a kangaroo word for SON
ALLOCATE is a kangaroo word for ALLOT
MERCER is not a kangaroo word for CEER
```

Problem 9 was removed from the contest. (prob9)

Nth Derivative (prob10)

The Problem

A function *f(x)* is a polynomial function of *x* if every term of *f(x)* can be represented as ax^n, where *a* and *n* are constants. An example of such a function is

$$f(x) = 5x^6 - 7x^{-3} + 16x + 8$$

Here is the same example function as it might appears in code:

```
f(x) = 5*x^6-7*x^-3+16*x+8
```

For such a function, its derivative, *f'(x)*, can be calculated term by term in the familiar way: the derivative of ax^n is anx^{n-1}. The derivative of a constant is 0. For our example, *f'(x)* is

```
f'(x) = 30*x^5+21*x^-4+16
```

We could repeat the process to calculate a second derivative, *f''(x)*, as

```
f''(x) = 150*x^4-84*x^-5
```

Given a polynomial with an arbitrary number of terms, calculate its *N*th derivative. Here are your guidelines

- You may assume that all constants (coefficients and exponents) will be 32-bit integers.
- Coefficients of 0 or 1 will not explicitly appear in the problem and should not appear in your solution (`x^5` is ok, `1*x^5` and `0*x^5` are not).
- Exponents of 0 or 1 will not appear in the problem and should not appear in your solution (`6*x` is ok, `6*x^1` and `6*x^0` are not).
- Zero constants should not appear in your solution unless the final result is zero.
- You do not have to reduce your answer by combining like terms.
- There should be no spaces in your output.

Input

The input file consists of an integer on line 1 indicating the number of test cases in the file. There are two lines of input for each test case. The first line in a test case is the desired derivative, a positive integer no greater than 10. The second line is a string representing a polynomial in *x*.

Output

Your output should have as many lines as there are test cases. Each line should contain the specified derivative of the test case polynomial.

Sample Input

```
2
2
5*x^6-7*x^-3+16*x+8
1
6*x^3+111
```

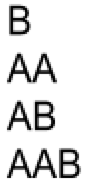
Sample Output

```
150*x^4-84*x^-5
18x^2
```

Schrödinger’s Character (prob11)

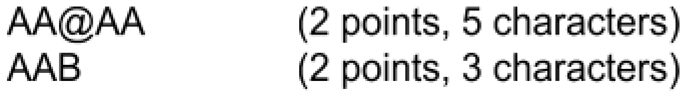
The Problem

Agent #015, of the Mars Exploration Research Commission Experimental Regiment, has been given an important new task. One of the Martian warlords has developed powerful new technology, and it is up to agent #015 to determine just how powerful this will be. As you may know, the Martians determine who is the strongest among them with a most peculiar contest. The eldest Martian decrees a list of K words.



B
AA
AB
AAB

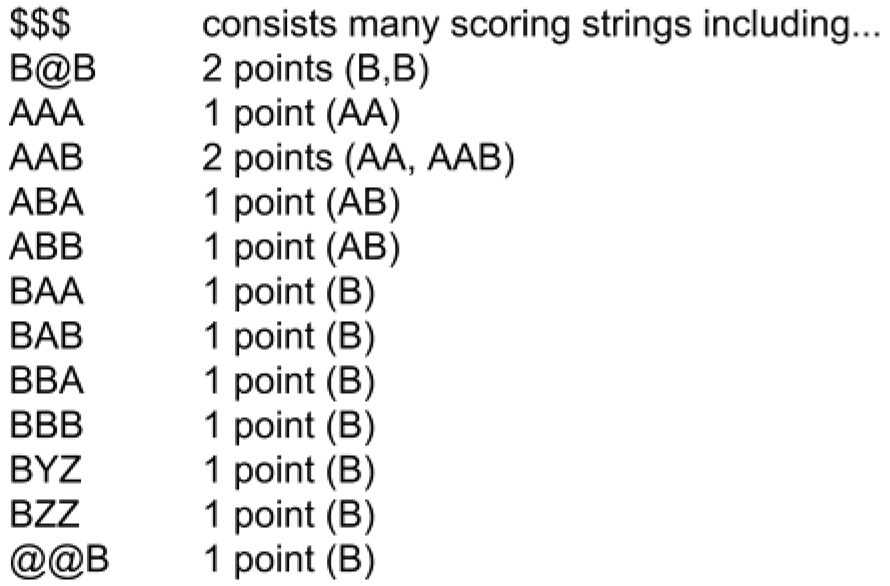
Then, the Martian warlords compete amongst themselves to spell as many of these words as possible using only X characters. As the Martians have not yet invented the newline, they separate all their words with the ‘@’ character. However, to make the contest more interesting, they have decided that a word can count for all prefixes that are matched.



AA@AA	(2 points, 5 characters)
AAB	(2 points, 3 characters)

Note that AAB only counts for 2 points (AA and AAB), as although AB is contained it is not a prefix.

However, the balance of power seems set to be thrown completely off, as one of the warlords has created the “Schrödinger’s Character” (represented by ‘$’). This character is simultaneously all other characters, even the ‘@’! Help Agent #015 to determine how many points this will let the warlord achieve. As this number may be quite large, calculate it mod M. In addition, to better evaluate the new balance of power, calculate the maximum score that can be achieved without use of Schrödinger’s Character. This second number should not be under mod, as the agency would like to directly compare this with their previous records.



$$$	consists many scoring strings including...
B@B	2 points (B,B)
AAA	1 point (AA)
AAB	2 points (AA, AAB)
ABA	1 point (AB)
ABB	1 point (AB)
BAA	1 point (B)
BAB	1 point (B)
BBA	1 point (B)
BBB	1 point (B)
BYZ	1 point (B)
BZZ	1 point (B)
@@B	1 point (B)

Input

Input will consist of T test cases. Each case will begin with the number K ($1 \le K \le 200$) representing the number of words to follow. The next K lines will contain a word comprised of the characters (‘A’-’Z’). There will be no more than 200 characters across all words for each test case. Following this will be the number X ($0 \le X \le 10^{12}$), the number of characters the warlords may use, and the number M ($2 \le M \le 10^{9}$), the mod to use.

Output

For each test case, output the maximum score achievable by using Schrödinger’s Character under mod of M, followed by the maximum score achievable without using Schrödinger’s Character, not under mod.

Sample Input	Sample Output

4
1
A
1 1000
1
A
2 1000
4
B
AA
AB
AAB
3 1000000
2
A
A
1 10000
	1 1
	28 1
	840 2
	2 2

Unfinished Crossword Puzzle (prob12)

The Problem

Given a crossword puzzle, without the hints, and some letters already specified along with a collection of the letters that are to be used to solve the puzzle, you are to find all ways that the puzzle can be completed using each letter in the collection exactly the number of times that it occurs in the collection.

Input

The first line contains an integer *M*, the number of cases/problems that are in the file.

Each case begins with an integer *N*, which is the number of subcases for this problem, followed by a string *s* that contains the letters to use to solve the crossword puzzle. The string *s* will be no longer than 15 uppercase letters. Each subcase begins with a pair of integers *r* and *c* which are the number of rows and columns respectively of the crossword puzzle. The next *r* rows each contain one row of the crossword puzzle. Empty space is marked by '.', a cell to be filled is marked by '_', and a predefined location will contain a capital letter. There will be at least four cells of the crossword puzzle filled in, but those values may change from subcase to subcase.

After the last subcase, the remaining lines contain the dictionary to use when solving the crossword puzzle. The number of words in the dictionary will not exceed 350, 000.

Output

Label the output for each problem by Problem 1, Problem 2, and so on. For each subcase, print all possible solutions, as a crossword, where the periods have been replaced by spaces. The possible solutions may be printed in any order. Each solution should be separated by a line of ='s (30 characters in length). Finally, at the bottom of each subcase, list the number of solutions that have been found using the format shown in the sample output. Refer to the sample output for the output formatting.

Sample Input

```
1
2 ATOHAVUETDAGETI
6 15
_..............
A_.............
___E_..........
_...A..........
....__U_.......
...._..........

7 6
._....
_E_...
._....
._.A__
.___O.
._....
.E....

aha
at
auto
ava
avoid
deviate
eat
eight
gave
get
godet
gut
heave
ho
oath
taut
tha
that
ti
to
video
```

Sample Output

```
Problem 1
==============================
g
at
divet
e   a
    hout
    a
==============================
g
at
video
e   a
    taut
    h
==============================
Subcase 1 has 2 solution(s).
==============================
 d
get
 v
 i aha
 auto
 t
 e
==============================
Subcase 2 has 1 solution(s).
==============================
```