MERCER UNIVERSITY

Spring Programming Contest

February 27, 2016

Class Stats	(prob1)
Magic Squares?	(prob2)
Friday the 13th	(prob3)
Gifts for the twins	(prob4)
DNA	(prob5)
FSM2	(prob6)
Rebel Blockade Sprinter	(prob7)
A Game of Stones	(prob8)
Lucas's Letters	(prob9)
Intergalactic Soccer	(prob10)
Unpacking a Mystery	(prob11)
Scrambled String Classes	(prob12)
1-800-MNEMONIC	(prob13)

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).

Class Stats (prob1)

The Problem

Professor Kerr has discovered that an average isn't always the best indicator of how his students have done on his exams. He is interested in developing a program that will display the *median* value of his students on his exams. The median value isn't impacted as much by outlier scores, those scores that lie very far below or above the rest of the class distribution. To calculate it, you merely find the middle value in an ordered distribution of the scores. If there are an even number of scores, it is the average of the two scores adjacent to the middle location. For example, if there are 26 scores, the median is the average of the thirteenth and fourteenth scores.

In addition to the median, he would also like to show the highest score on the exam as well as the number of As, Bs, Cs, Ds, and Fs. An A is defined as a score greater than or equal to 90, a B is a score greater than or equal to 80, a C is a score greater than or equal to 70, a D is a score greater or equal to 60, and an F is a score less than 60.

Input

The input will be one or more lines. Each line will begin with a single nonnegative integer value representing a student's exam score on a scale from 0 up to 100. The end of input will be triggered by a negative 32-bit integer.

Output

The output will be seven lines as shown below. The first line of output consists of the median of the students' scores preceded by `Median = `. Include exactly one digit after a decimal point followed by a % sign. The second line consists of the highest score preceded by `High Score = `. This is followed by five lines of output with the breakdown of the letter grade frequencies. Each frequency should be preceded by a header like `As = `.

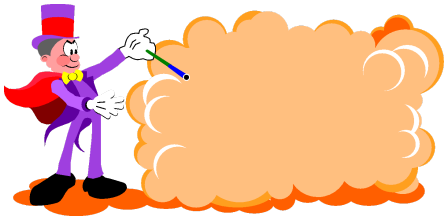
Sample Input

```
87
96
85
25
42
41
100
82
85
86
78
79
-1
```

Sample Output

```
Median = 83.5%
High Score = 100
As = 2
Bs = 5
Cs = 2
Ds = 0
Fs = 3
```

Magic Squares? (prob2)



The Problem

A two-dimensional square array is *magic* if the sum of every row, column, and both diagonals all match. In the following example, this square array with three rows and columns is indeed magic since all rows, columns, and diagonals sum to 15.

8	**1**	**6**
3	**5**	**7**
4	**9**	**2**

Input

The first line of input contains a single integer n, ($1 \leq n \leq 1000$), which is the number of test matrices that follow. For each matrix being tested, the input will be a single line containing one integer representing the number of rows r in the range between 1 and 10 inclusive, ($1 \leq r \leq 10$). Since this is a square array, you will always have an equal number of rows and columns. This is followed by r rows each of r integers each separated by a space. All input is in row major order. Each integer and each magic sum are valid 32-bit integers.

Output

For each matrix, output a single word `yes` or `no` in lowercase indicating whether it is magic.

Sample Input

```
2
3
8 1 6
3 5 7
4 9 2
4
16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1
```

Sample Output

```
yes
yes
```

Friday The 13th (prob3)

The Problem

It can be shown that there is at least one Friday the 13th each year under the Gregorian calendar. However, there can be more. In 2015, for example, there are three such days: February 13, March 13, and November 13.

Your job is to calculate the total number of days that fall on Friday the 13th in a range of years under the Gregorian calendar.

You need to take into consideration the leap years. The month of February in a leap year has 29 days. Every year that is exactly divisible by 4 is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the years 2015 and 1900 are not leap years, but the years 2016 and 2000 are.

Input

The input consists of a number of test cases. Each of the input lines contains two integers representing a test case:

```
year1 year2
```

where 1582 < *year1* <= *year2* < 300000000000. The search for Friday the 13th is conducted on the years from *year1* to *year2* including *year1* and *year2*. The input is terminated by a line consisting of 0 0.

Output

For each input case, print the total number of days that fall on Friday the 13th in the years given by the interval [*year1*, *year2*].

Sample Input

```
2015 2015
2016 2016
2000 2015
0 0
```

Sample Output

```
3
1
28
```

Gifts for the twins (prob4)

Time limit: 15 seconds
Memory limit: 256 megabytes

The Problem

Twin little brothers Coder and Tester still haven't unpacked their Christmas presents. The presents that they got are arranged in a line. Brothers will unpack the presents at the same time one by one. Coder will start from the left and his brother from the right. They will meet in the middle, and if there is only one present left, neither of the brothers takes it and they donate it to their favorite dog. Each present has a price tag. When the brothers finish unpacking, they will compare the total price of their presents, and if it's different an immediate Armageddon will follow. To avoid this unfortunate turn of events, their mother has decided to remove a subset (possibly empty) of presents from the line. She still wants Coder and Tester to be happy, therefore she wants the total price of presents each brother will get to be the maximum possible. You also want to avoid the Armageddon, so please help mother to determine the maximum possible total price of gifts a brother can get. Of course, leaving the brothers with zero presents isn't the end of the world.

Input

The input starts with a single integer T ($1 \le T \le 50$) - the number of test cases. Each test case starts with a number n ($1 \le n \le 50$), the total number of presents in the line. The next line will contain n numbers $0 \le a_i \le 20$ - the prices of the presents in order.

Output

For each i-th (1-based) test case output "Case i: "without quotes followed by a single integer - the maximum possible total price of presents Coder and Tester can each get.

Sample Input

```
5
2
5 5
4
3 8 3 2
4
8 4 5 7
5
1 2 3 4 5
10
1 4 9 1 1 1 14 1 3 1
```

Sample Output

```
Case 1: 5
Case 2: 3
Case 3: 12
Case 4: 0
Case 5: 3
```

DNA (prob5)

Timelimit: 1 second

As everyone knows New Horizon found alien life forms on Pluto. What the public does not know is that the new species has had its DNA sequenced. What is known so far is that all found species have only two types of nucleotides, commonly referred to as A and B. Also known is that for every A there is a corresponding unique B that can be found later in the DNA's sequence, and for every B a corresponding unique A exists earlier in the sequence. This means that every sequence is of even length. Moreover if a sequence has *2n* nucleotides, then it must contain exactly *n* A's and *n* B's.

Scientists have been arguing about how many possible species can exist, and they are requesting your help to write a program to help check their math.

The Problem

Given an *n* and *m*, please determine the number of possible sequences of length *2n* modulo *m*. This will allow the scientists to check their values.

Input

The input will begin with a positive integer c, ($c \leq 500$), indicating how many test cases will be evaluated.

Each test case will contain 2 space separated integers on a single line. The first integer will represent n ($1 \leq n \leq 50{,}000$), the number of A and B nucleotides. The second integer will represent m ($1 \leq m \leq 1000$), the modulo used for determining the number of sequences of length $2n$.

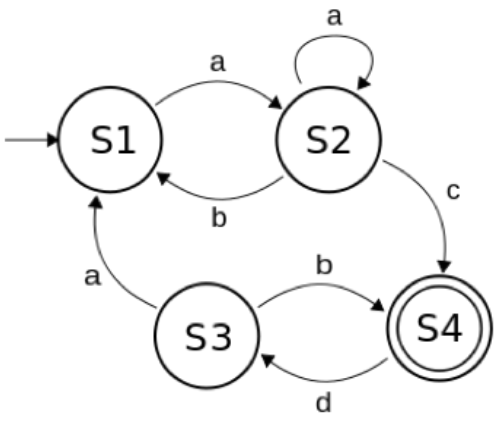
Output

For each test case output the number of sequences of length $2n$ modulo m.

Sample Input	Sample Output
4 1 10 2 1000 4 7 4 10	1 2 0 4

FSM2 (prob6)

The Problem



A finite state machine is a model of a computational system that consists of a set of states, an alphabet of symbols that serves as a set of possible inputs to the machine, and a transition function that maps each state to another state (or to itself) for any given input symbol. One state is designated as the *start state*. One or more states can be designated as an *accepting state*. The machine operates by being fed a string of symbols that makes it move through a series of states. The machine "accepts" the string if the moves end at an accepting state. The figure at the right shows a finite state machine that has four states {S1, S2, S3, S4}. The alphabet contains four symbols (**a**, **b**, **c**, and **d**). The start state is S1. The only accepting state is S4, indicated by the double circles. The transitions are provided by the arrows.

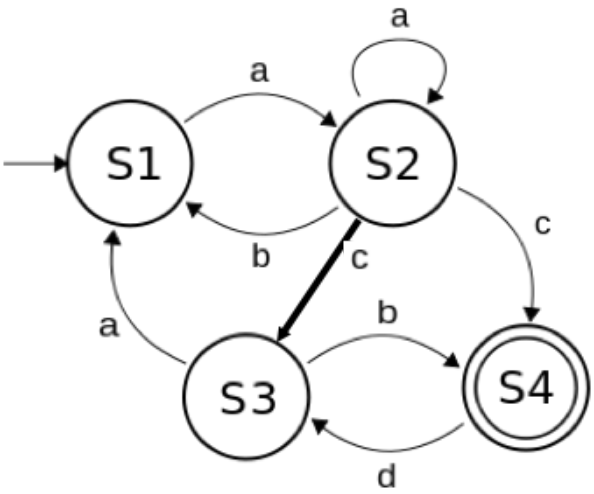
If we feed this machine the string **`aacdbdaac`**, then it will move through states S1→S2→S2→S4→S3→S4→S3→S1→S2→S4. The machine accepts this string because the sequence of moves ends in S4, an accepting state. The machine does not accept the string **`acd`** because the sequence of moves ends in S3, which is not an accepting state. The string **`abc`** is not accepted either because there is no transition from S2 when the input symbol is **`b`**.

We can use a table to define the transitions as illustrated below for the FSM shown above..

In a well-formed FSM, a given symbol can be associated with at most one transition beginning in any given state. However, let's relax that restriction in an *FSM2*. Suppose that we can have two or more transitions from a state for the same symbol. For example, we might add a second transition [shown on the next page with a heavier arrow] for the symbol **`c`** from S2 as shown below. The table for the transitions is similar to the one above, but has an extra row for the transition from S2 to S3 when the input is **`c`**. However, transitions out of a state cannot be duplicated—that is, no two rows of the transition table can be the same. For example, there can't be a second transition from S2 to S4 labeled by **`c`**.

From state	Symbol	To state
S1	**`a`**	S2
S2	**`a`**	S2
S2	**`b`**	S1
S2	**`c`**	S4
S3	**`a`**	S1
S3	**`b`**	S4
S4	**`d`**	S3

The ambiguity concerning which transition to make [which transition should be used?] is resolved by cycling through the transitions in the order they appear in the table. So, for example, the first time the machine above is in state S2 and **c** is the next symbol in the input string, then the machine transitions to S4. The second time the machine is in state S2 and **c** is the next symbol in the input string, then the machine transitions to S3.



From state	Symbol	To state
S1	**a**	S2
S2	**a**	S2
S2	**b**	S1
S2	**c**	S4
S2	**c**	S3
S3	**a**	S1
S3	**b**	S4
S4	**d**	S3

Input

The input consists of a sequence of test cases, each representing a scenario involving a finite state machine and a sequence of input strings to be processed by the machine.

The states in an FSM2 are designated using contiguous positive integer values starting at 1: 1, 2, …, *N*, where *N* is the number of states. State 1 is the start state. Symbols are a subset of the lower-case letters of the English alphabet, **a**, **b**, **c**, …, **z**.

Each test case occupies (*M* + *K* + 3) lines, where *M* and *K* are defined below.

- The first line contains a number *M* of transitions in the FSM2 for the test case, 0 ≤ *M* ≤ 256. If M is zero, there are no more test cases. If M > 0, then the next *M* lines provide the entries in a transition table, one row per line. Each line contains three values: a from state (an integer in the range 1 .. *N*), a symbol (one of **a**, **b**, **c**, …, **z**), and a to state (an integer in the range 1 .. *N*). The values are separated by a single space character.
- The next line contains the accepting states, a sequence of positive integer values with space separators.
- The next line contains a number *K* (1 ≤ *K*), the number of input strings to feed the FSM.
- The next K lines each contains a string of lowercase English letters. A string contains between zero and 1024 letters, inclusive.

Output

The output is a sequence of lines for each test case. The first line for each test case is FSM2 followed by a space, followed by an octothorp (#), followed immediately by the test case number. Each of the next *K* lines shows whether a string is accepted by the machine. The strings appear in the output in the same order as given in the input. Each line of output is the input string delimited by quotation marks followed by a single space followed by either ACCEPTED or NOT ACCEPTED. The word END should appear after all test cases have been processed–that is, when the value of *M* in the input is 0.

Sample Input

```
8
1  a  2
2  a  2
2  b  1
2  c  4
2  c  3
4  d  3
3  b  4
3  a  1
4
5
aacdbdaac
acd
abc
ac
aaacdaacaac
4
1 a 2
2 a 3
3 a 2
2 a 1
2 1
9

a
aa
aaa
aaaa
aaaaa
aaaaaa
aaaaaaa
aaaaaaaaaaa
0
```

Sample Output

```
FSM2 #1
"aacdbdaac" NOT ACCEPTED
"acd" NOT ACCEPTED
"abc" NOT ACCEPTED
"ac" ACCEPTED
"aaacdaacaac" ACCEPTED
FSM2 #2
"" ACCEPTED
"a" ACCEPTED
"aa" NOT ACCEPTED
"aaa" ACCEPTED
"aaaa" ACCEPTED
"aaaaa" ACCEPTED
"aaaaaa" NOT ACCEPTED
"aaaaaaa" ACCEPTED
"aaaaaaaaaaa" ACCEPTED
END
```

Rebel Blockade Sprinter (prob7)

The Problem

The evil empire has developed a dastardly weapon that has the potential of destroying the entire galaxy. They plan to use this weapon to spread their influence and thwart the hopes of the rebels trying to overthrow their tyrannical rule. The rebels have managed to purloin a copy of the operational schematics of the weapon. After analyzing the plans extensively, it is determined that the entire weapon can be destroyed if a high energy explosive can be detonated at the core of the weapon.

However, the only way to get to the core is through a series of tunnels, which can be entered through an air vent. The base is designed in such a way, that from each air vent there is only one path to the core of the weapon. Air vents and intersection of the tunnels are guarded by turrets. Each turret is sufficiently powerful that it will prevent any and all rebel troops that attempt to pass though that turret location, but the rebels still have hope. The rebels know exactly how expensive it is to activate the turrets.

The rebels have determined if they steal enough resources from the empire, they will be unable to power on the turret to defend against a full frontal assault. To help the rebels determine if is safe to assault the weapon, determine the minimum cost it would take the empire to stop the rebels from reaching the core. You are their only hope.

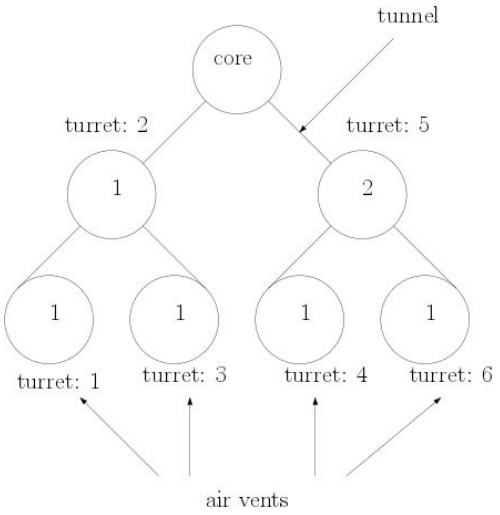


Image corresponds to the first example from the sample input
The possible solutions are to select turrets 2 and 5 or to select turrets 2, 4, and 6

Input

An integer, N, representing the number of test cases to follow(1<=N<=10).

Each test case will consist of:
An integer, T, the number of turret locations(1<=T<=1,000,000).

T lines containing three positive integers -- i, c, and n -- representing turret id number(1<=i<=T), the cost to activate that turret (0<=c<=4000), and the id of the next turret on the path to the core or 0 if the core is the next location respectively (0<=n<=T).

Output

N integers representing the minimum cost to stop the rebels for the nth test case.

you may assume all integers will fit in a 32 unsigned bit integer or a long if you are using java

Sample Input

```
2
6
1 1 2
2 1 0
3 1 2
4 1 5
5 2 0
6 1 5
3
1 4 0
2 1 1
3 2 2
```

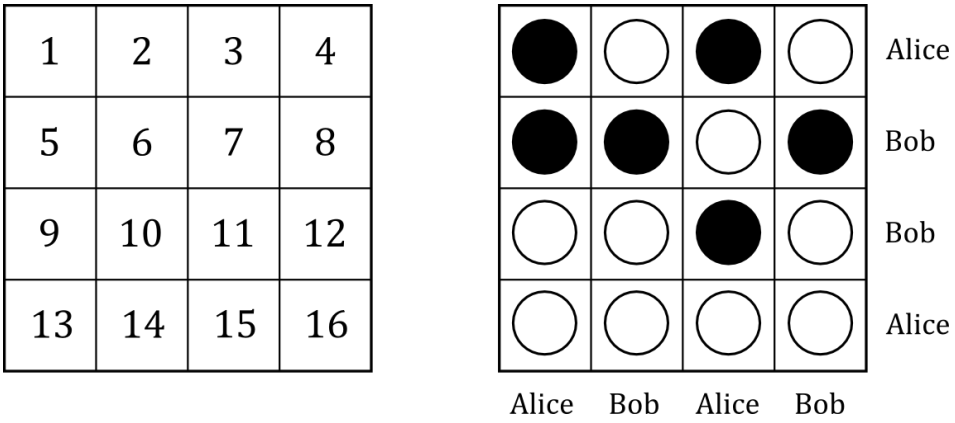
Sample Output

```
3
1
```

A Game of Stones (prob8)

The Problem

Alice and Bob are playing a game on a grid. In this game Alice and Bob alternate taking turns, placing one stone per turn. Alice starts first. Initially the grid is empty and they start in the upper left cell. During each turn the player has a choice, place a single black stone in the active cell or single white stone in the active cell. After a turn is complete, the active cell becomes the next cell in row major order. The diagram to the bottom left shows the transition of active cells throughout the game.



After the game is complete, the grid is scored. Alice gets a single point for each row and column that contains an even number of black stones. Bob gets a single point for each row and column that contains an odd number of black stones. The diagram above to the right shows an example of a completed board and the rows and columns that Alice and Bob win, respectively.

Given that Alice and Bob both want to maximize their points, how many points will Alice and Bob win if both players play optimally?

Input

The input starts with a single integer t representing the number of games to analyze.

The next t lines contains two integers r and c $(1 \le r, c \le 1000)$ representing the number of rows and number of columns in the grid.

Output

For each case, output the header "`Case #d:`" where d is the current case number starting from 1. On the next line output "`Alice: a, Bob: b`" where a and b are the number of points earned by Alice and Bob, respectively. Follow the output of each case with a blank line.

Sample Input

```
6
1 1
3 1
4 4
3 9
3 2
996 887
```

Sample Output

```
Case #1:
Alice: 2, Bob: 0

Case #2:
Alice: 2, Bob: 2

Case #3:
Alice: 2, Bob: 6

Case #4:
Alice: 6, Bob: 6

Case #5:
Alice: 1, Bob: 4

Case #6:
Alice: 941, Bob: 942
```

Lucas’s Letters (prob9)

Lucas loves playing word games, but because he is an only child, he doesn't have someone to play against often. Luckily, he's found a phone app that has a one player mode. In the game, he's given a set of tiles, where each tile has a letter and a corresponding score. Unlike other word games, tiles with the same letter may have different scores. In the game, Lucas's goal is maximize his score by creating distinct words using the given tiles, where each tile may be used for only one word. For example, if the set of tiles contains one c, two a's, one t and one b, Lucas could form the word "cat", but he could not form both "cat" and "bat" because there is only one tile with a t and that one tile can't get used for both "cat" and "bat". Lucas's score at the end of the game is simply the sum of the scores of tiles used in the words he forms minus the sum of the scores of the leftover tiles.

The Problem

Given a list of tiles available to Lucas, as well as each word in Lucas's limited vocabulary, determine the maximum score he could achieve in the word game.

Input

The first line of input will consist of a single positive integer, n $(n \leq 100)$, representing the number of input cases. The first line of each input case will consist of a single positive integer, t $(t \leq 200)$, representing the number of tiles available for the input case. The following t lines will each contain a single lowercase letter followed by a space, followed by a positive integer, representing a tile for the input case. Each of the scores of the tiles will be in between 1 and 1000, inclusive. The next line of each input case will contain a single positive integer, w $(w \leq 16)$, representing the number of words in Lucas's vocabulary for the input case. The following w lines will contain one word each, in lowercase letters only, representing a word in Lucas's vocabulary for the input case. Each of these words will be in between 1 and 15 letters long, inclusive.

Output

For each input case, output a single integer on a line by itself representing Lucas's maximum possible score for that input case.

Sample Input

```
3
4
c 3
a 1
t 1
b 4
2
bat
cat
1
i 17
1
i
10
c 10
c 20
c 30
c 40
e 17
e 8
a 1
k 5
p 15
u 35
4
tree
cup
cream
cake
```

Sample Output

```
3
17
105
```

Intergalactic Soccer (prob10)

Intergalactic Soccer is a video game where the offensive team must choose a sequence of passes followed by a shot on goal, on a single possession. The ball always starts with player number 1, and any player may take the shot on goal. The probability for each player making a shot on goal is known. Furthermore, each player can pass to a few teammates and the probability that those passes are successful is known as well. Once a shot on goal is taken or a pass is unsuccessful, the ball goes to the other team and the possession is ended.

The Problem

Given each player's probability of making a shot on goal and each player's probability of making a successful pass to other players, calculate the maximum probability of the team making a shot, if they choose the best sequence of passes followed by a shot on goal, on a single possession.

Input

The first line of input will consist of a single positive integer, *n (n ≤ 30)*, representing the number of input cases. The first line of each input case will consist of a single positive integer, *p (p ≤ 10000)*, representing the number of players on the team, for the input case. The following line will contain *p* space-separated real numbers, $x_1, x_2, \ldots x_p$, in between -11 and 0, inclusive, to no more than four decimal places, where 2^{x_i} represents the probability of player number *i*, making a shot on goal ($1 \le i \le p$). The next line of input will contain a single non-negative integer, *e (e ≤ 40000)*, representing the number of possible passes that can be made on the team for the input case. The following *e* lines will each contain information about a possible pass. Each of these lines will contain two space-separated integers, *a (1 ≤ a ≤ p)* and *b (1 ≤ b ≤ p, b ≠ a),* followed by a space and a real number, *x*, in between -1 and 0, inclusive, to no more than four decimal places, where 2^x represents the probability that player a can successfully make a pass to player b. ***Note that if player a can make a pass to player b, it is possible that player b can not make a pass to player a, or that the probability of success of such a pass may be different.***

Output

For each input case, output a single real number, rounded to 3 decimal places, representing the maximum probability of scoring on a single possession, if the appropriate sequence of passes and shot are selected.

Sample Input	Sample Output
2 5 -.4 -.2 -.3 -.2 -.1 3 1 2 -.05 1 3 0 2 5 -.08 2 0 -1 1 1 2 0	0.853 1.000

Unpacking a Mystery (prob11)

The Problem

While exploring an ancient ruins, you've discovered a mysterious encrypted text. After inspection, you see that the text is comprised only of 4 characters: '#', '$', '0', and '1'.

Additionally, you found a guide that explains how the encryption was formed. Let f(S) denote the encryption of the string S. First, the encryption only applies to binary strings S of length 2^n for some n>0. Then, if the string is all the same character, return that character as the encryption. Otherwise, recurse on each half of the string; say the first half of the string is S_1 and the second half is S_2. Finally, concatenate the encryption as #f(S_1)$f(S_2). You would like to reverse this encryption to recover the original string.

However, there is a potential complication. The decrypted string might be very large and therefore impossible to store on a standard computer. So instead of fully decrypting the string, you decide instead to develop the following functionality. Given the encrypted string, answer queries of the form: is the bit at a specified position in the decrypted string a 0 or 1? (the first character of the string is considered position 0 and the last character is position 2^N-1)

Input

The first line of the input contains a single integer 1<=T<=10, the number of test cases. Then, T test cases follow.

For each test case, the first line contains two integers N and Q. N denotes that the original string S has length 2^N, and Q is the number of queries. You are guaranteed 1<=N<=1,000 and 1<=Q<=10,000.

The second line of the test case contains the encrypted string. Recall that this string will only contain 4 different characters: '#', '$', '0', and '1'. The length of the encrypted string is at most 100,000.

Then, Q lines follow. The queried position will be given to you in binary, where each binary string is of length exactly N.

Output

For each test case, output the query answers on a single line, where each query is separated by a space. Each query answer should be a single 0 or 1.

Sample Input

```
3
1 2
1
0
1
2 4
##1$0$1
00
01
10
11
3 5
#0$##0$1$0
011
001
101
111
110
```

Sample Output

```
1 1
1 0 1 1
0 0 1 0 0
```

Sample Input/Output Explanation

There are 3 test cases. For the first test case, the original string is of length 2, and the encryption yielded a single “1”. Thus, the decrypted string is “11”. For the second test case, the decrypted string has length 2^2=4, and the original string is 1011. For the third test case, the original string has length 2^3=8, and the original string is 00000100.

The output is then the bits at the queried positions in the original string.

Scrambled String Classes (prob12)

The Problem

Qwrgly, a linguist on planet Zxcvby, has been assigned the task of determining how many "equally scrambled" groups of strings of length ***n*** can be generated from an alphabet of ***m*** characters.

Two strings ***x*** and ***y*** of equal length are "equally scrambled" (***x*** ***y***) if the following two conditions apply:

1. (x_i != x_j) *if and only if* (y_i != y_j)
2. (x_i = x_j) *if and only if* (y_i = y_j)

where x_i is the i^{th} character of the string ***x***.

So, for ***m***=2, ***n***=3, there are 2^3 strings: "aaa", "aab", "aba", "abb", "baa", "bab", "bba", "bbb" (assuming a and b are characters in Qwrgly's alphabet). The strings "aaa" and "bbb" are equally scrambled, as are "aab" and "bba", "aba" and "bab", and "abb" and "baa". Thus, there are four **equivalence classes** of equally scrambled strings for ***m***=2, ***n***=3.

Since m^n can get large very quickly, Qwrgly needs your help – he can't possibly construct all the strings to determine the number of equivalence classes. Help him determine the number of equivalence classes of equally scrambled strings of length ***n*** using an alphabet of ***m*** characters.

Input

The first line is an integer ***t*** where 1 <= ***t*** <= 100, denoting the number of test cases.
The next ***t*** lines are test cases described in a single line with two integers ***m*** and ***n***, with 1 <= ***m, n*** <= 2000.

Output

For each test case ***m n***, print the number of equivalence classes of scrambled strings of length ***n*** from an alphabet of ***m*** characters. Since the number can be large, you must print it modulo **$(10^{18} + 7)$**.

Sample Input

```
6
1 1
1 2000
2 2
2 3
20 25
950 1500
```

Sample Output

```
1
1
2
4
638590332058238819
74093576401043362
```

1-800-MNEMONIC (prob13)

Full disclosure/note to the curious: this problem is an adaptation of an older problem that featured in several related studies done in 1999 and 2000 to compare program [and programmer] efficiency across multiple programming languages. This variant features a few new twists and a completely different backstory. You can read more about the original version at http://www.norvig.com/java-lisp.html.

The Problem

The Tell Belaphone and Belagraph Communication Consortium (better known as "Pa Tell"), inventors and sole patent holders on all long-distance communication technologies in the venerable Queendom of Belaria, wish to provide a new service to their highest-paying business subscribers. Since Belarians are famously bad at remembering the long sequences of digits that identify belephone terminals (better known as "'phone numbers"), Pa Tell has invented a mnemonic mapping system that replaces digits with letters. If the letters spell meaningful words or phrases, phone numbers will be much easier for the poor Belarians to remember.
For example, the hard-to-remember phone number (516) 639-3864 could map to the much friendlier mnemonic phrase Hello World. Or it could map to Fellow Urk'd. [1]
To promote this new feature of the Pa Tell system (and to encourage brisk sales of the service), Pa Tell wishes to make it as easy as possible for customers to see how their existing business phone numbers could be mapped to appropriate mnemonic phrases. To this end, you have been hired to write an automated translation program.
For a given phone number (a sequence of digits that may or may not contain spaces, hyphens, and/or parentheses) and a given dictionary (a list of words), your program must produce a list of all possible mnemonic phrases, where "possible" is defined by the following rules:

- Digits are mapped to letters according to the following table:

Letters	Digit
e	1
ai	2
ou	3
bcd	4
fgh	5
jkl	6
mnp	7
qrs	8
tvw	9
xyz	0

- Non-digit characters in phone numbers are *ignored* for the purposes of mnemonic mapping, but are *retained* when printing output
- Likewise, non-letter characters in dictionary words are *ignored* (as are case differences) for the purposes of mnemonic mapping, but are *retained* when printing output
- When printing mnemonic phrases, separate each word with a space
- Words *can* occur multiple times in the mnemonic phrase
- The words of a mnemonic phrase must match the phone number digits *completely* (i.e., have no gaps of unmatched digits or extra letters "hanging off" the end)

- If multiple words could match the number at a given location, your program should try all of them from *longest* to *shortest*; if multiple potentially matching words are the same length, your program should try them all in *alphabetical order*
- Special cases:
  - If your program comes to a point in the phone number where *no* dictionary word matches, it should insert **one** literal digit into the output mnemonic phrase and continue searching for word matches starting at the next digit **if and only if**
    1. This is neither the beginning nor the end of the mnemonic phrase (i.e., the mnemonics 2 Good To Be True and Good 1 are impossible)
    2. No other digits have been inserted into the mnemonic phrase (i.e., the mnemonic Oops 2 Fast 2 Furious is impossible, but Set 4 Life is allowed)
  - If your program comes to a point in the phone number where *no* dictionary word matches *and* the remaining digits follow the pattern N000..., where N is any non-zero digit and 000... is a sequence of one or more zeros, it should terminate the current mnemonic phrase with the N000... sequence itself **if and only if** this is not also the beginning of the phrase (i.e., the mnemonic 1000000 is impossible, but the mnemonic Enigma Philosophy Opera 4000 is allowed)

Input

Input consists of one or more data sets, each consisting of a *dictionary* followed by a *phonebook*. Each data set must be processed in complete isolation from the others (i.e., each new dictionary completely *replaces* the previous one, it does not *augment* it).
A *dictionary* consists of a list of dictionary words, one per line. It is terminated by a blank line.
A *phonebook* consists of a list of phone numbers, one per line. It is terminated either by EOF (at which point the program should terminate) or by a blank line, which indicates that another data set is coming.

Output

For each phone number in a phonebook, your program should print one line of output for each possible mnemonic phrase in the following format:

```
(516) 639-3864: Fellow Urk'd
```

The output from different data sets should be *separated* by a single empty line, but there should be no trailing empty line at the end of all output.

1. The "Urk'den" (sing. "Urk'd") are a race of heroic but rather dense rock-people that feature prominently in Belarian mythology and folklore.↩

Sample Input

```
Hello
World!
World$
Hello_World
FELLOW
urk'd
Set
Life
good
ne
One
Gateway
ax
zz

(516) 639-3864
819 4 6251
533.4071
533.80.711
52,919,202,000
52,919,202,00
52,919,202,0

Gateway
ax

52,919,202,000
52,919,202,00
52,919,202,0

Gateway

52,919,202,
52,919,202,0
52,919,202,00
52,919,202,000

bob
```

Sample Output

```
(516) 639-3864: Hello_World
(516) 639-3864: FELLOW urk'd
(516) 639-3864: Hello World!
(516) 639-3864: Hello World$
819 4 6251: Set 4 Life
533.4071: good 0 ne
52,919,202,000: Gateway ax zz
52,919,202,0: Gateway ax

52,919,202,0: Gateway ax

52,919,202,0: Gateway 20
52,919,202,00: Gateway 200
52,919,202,000: Gateway 2000

(there is one blank line since
the last dataset does not contain
any phone numbers.)
```