Problem A

ONE-HANDED TYPIST

Input file: `typist.in`
Output file: `typist.out`
Java class name: `Typist`

Recently, a programming contest coach had a terrible accident and broke his left arm. It was then that he discovered how difficult it was to type with only one hand. He really needed to find a way to increase his typing speed, since he had to create problems in a hurry for an upcoming programming contest.

His solution was to switch from the standard QWERTY keyboard layout to a right-handed Dvorak keyboard layout (see diagrams below). It is pretty easy on a modern machine to change the keyboard layout in software, so that striking a key on the standard keyboard would give you the character at that position on the desired layout, whether or not it matches the physically labeled character.

The standard QWERTY keyboard layout, shift-modified on right.

The right-handed Dvorak keyboard layout, shift-modified on right.

One day, as the coach was typing out a problem statement with his right hand, he discovered that he had forgotten to change the keyboard layout from QWERTY to Dvorak on the machine! Naturally, all the text came out scrambled, because he had struck the keys as if it were a right-handed Dvorak keyboard. Alas, it would take too much time to retype all the text with only one hand.

Can you write a program to help the coach salvage his text?

Program Input

The input file consists of several lines of text, representing the text typed by the coach on a QWERTY keyboard layout when he thought he was using a right-handed Dvorak layout. Each line contains no more than 1000 alphanumeric characters, spaces, and punctuation found on the keyboard.

Program Output

For each line in the input, your program should write the line of text that the coach had meant to type. In other words, output the text as if it had been keyed using a right-handed Dvorak layout.

Sample Input & Output

INPUT	OUTPUT
Hg66t Mty6k! Jhg 4ibl; pytmn 8tc 5i79urrr t,gy jhg 6fxo kt.r	Hello World! The quick brown fox jumps... over the lazy dog.

Problem B

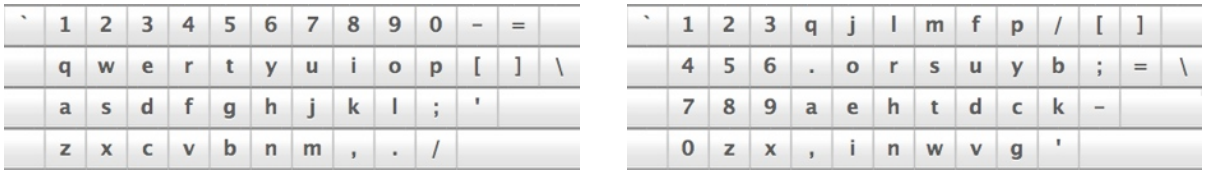
KEYBOARD COMPARISON

Input file: `keyboard.in`
Output file: `keyboard.out`
Java class name: `Keyboard`

After spending several weeks typing using the right-handed Dvorak keyboard layout, the coach (introduced in the previous problem) started to wonder if it really is any better. He decided to devise a method to verify this.

The coach would like to compare typing on the right-handed Dvorak keyboard against using a QWERTY keyboard with two hands and with one hand. A crude, but effective way to compare keyboard layouts is to measure the total distance the typing fingers must travel in order to type a certain passage of text. Can you write a program to help the coach perform this experiment?

The distance travelled by a finger will be measured from the centre of its home key to the centre of the target key and back. In order to simply to computations, we pretend that every key is a perfect square with unit length sides, and that all keys are laid out on a perfect grid (even though in reality they are usually staggered). The diagrams below show the straightened keyboards; see the previous problem for shift-modified diagrams.



The QWERTY and right-handed Dvorak keyboard layouts arranged on perfect grids.

On the QWERTY keyboard, the home keys are "ASDFJKL;" for two hands and "FGHJ" for one hand. On the Dvorak keyboard, the home keys are "EHTD". A key is always struck by the finger from the nearest home key.

For example, the distances for 'S' would be 0, 4, and 2 for the two-hand QWERTY, one-hand QWERTY, and Dvorak keyboards, respectively. Likewise, the distances for 'C' would be 2, 2.828427..., and 2, respectively.

Given a line of text, your task is to compute the total distance travelled by the typing fingers in typing that text, first using the QWERTY keyboard with two hands, then with one hand, and finally using the right-handed Dvorak keyboard. You can neglect any distance travelled in typing spaces or shifting.

Program Input

The input file consists of several lines of text. Each line contains no more than 1000 alphanumeric characters, spaces, and punctuation found on the keyboard.

Program Output

For each line of input, your program should write a line containing three real numbers accurate to 2 decimal places, representing the total distances travelled by the typing fingers. The first number corresponds to using the QWERTY layout with two hands, the second to one-handed QWERTY, and the third to right-handed Dvorak.

Sample Input & Output

INPUT	OUTPUT
CCPC Which keyboard is better? Numbers! 1234567890	8.00 14.81 10.47 40.14 60.24 39.43 58.60 74.72 92.52

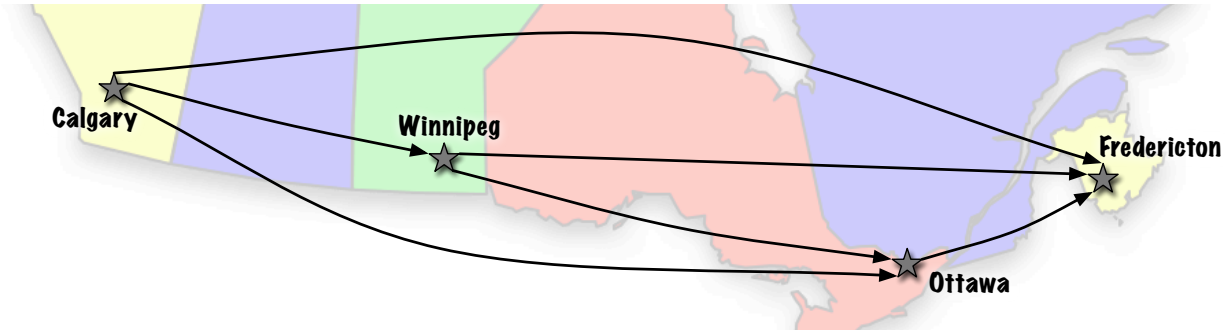
Problem C

FLYING TO FREDERICTON

Input file: `flying.in`
Output file: `flying.out`
Java class name: `Flying`

After being inspired by Salvador Dali's artwork, Brett decided he would like to travel to Fredericton, New Brunswick to visit the Beaverbrook Art Gallery.

Brett lives in Calgary, and would like to find the cheapest flight or flights that would take him to Fredericton. He knows that a direct flight from Calgary to Fredericton, if one exists, would be absurdly expensive, so he is willing to tolerate a certain number of stopovers. However, there are multiple airlines in Canada with so many different flights between different cities now, which makes it very difficult for Brett to find the least expensive way to Fredericton! Can you write a program to help Brett plan his route?



Map showing a sample of the flights that would take Brett to Fredericton.

You will be given a list of cities between and including Calgary and Fredericton. The cities will be given in order of "nearest" to "farthest". The first city will always be "Calgary" and the last "Fredericton".

You will also be given a list of flights between pairs of cities, and the associated cost for each flight, taxes included. There will never be a flight from a farther city to a nearer city — Brett has already discarded those flights, deeming them to be a waste of time and money. Bear in mind, however, that there may be more than one flight between any two cities, as Brett is considering flights from all airlines.

Finally, you are presented with a number of queries. Each query is a single integer indicating the maximum number of stopovers Brett will tolerate. For each query, your program must calculate the least total cost of flights that would take Brett from Calgary to Fredericton with no more than the requested number of stopovers.

Program Input

The first line of input contains a single number indicating the number of scenarios to process. A blank line precedes each scenario.

Each scenario begins with a number N ($2 \le N \le 100$), the number of cities, followed by N lines containing the names of the cities. A city name is a string of up to 20 upper- and lowercase letters. Next is a number M ($0 \le M \le 1000$), the number of flights available, followed by M lines describing the flights. Each flight is described by its departure city, its destination city, and an integer representing its cost in dollars. The final line starts with Q ($1 \le Q \le 10$), the number of queries, followed by Q numbers indicating the maximum stopovers.

Program Output

For each scenario, your program should output the scenario number, followed by the least total cost of the flights for each query. Follow the format of the sample output. If no flights can satisfy a query, write "No satisfactory flights". Output a blank line between scenarios.

Sample Input & Output

INPUT	OUTPUT
2 4 Calgary Winnipeg Ottawa Fredericton 6 Calgary Winnipeg 125 Calgary Ottawa 300 Winnipeg Fredericton 325 Winnipeg Ottawa 100 Calgary Fredericton 875 Ottawa Fredericton 175 3 2 1 0 3 Calgary Montreal Fredericton 2 Calgary Montreal 300 Montreal Fredericton 325 1 0	Scenario #1 Total cost of flight(s) is $400 Total cost of flight(s) is $450 Total cost of flight(s) is $875 Scenario #2 No satisfactory flights

Problem D

TRIANGULAR PEGS IN ROUND HOLES

Input file: `pegs.in`
Output file: `pegs.out`
Java class name: `Pegs`

Ever since we were small children, we have learned that square pegs do not fit into round holes. This is so much the truth that it has become a figure of speech in the English language. However, very little is ever said about triangular pegs. Have you ever wondered whether or not they would fit into round holes?

In this problem, you are given a description of a set of triangular pegs and round holes. You are to determine whether or not these triangular pegs would fit through the round holes.

Each peg is a perfect triangular prism where the lengths of the three sides of the triangular face are given. Each hole is a perfect circle with its diameter given. All pegs are longer than the diameter of the largest hole.

Program Input

The first line of the input file contains a number M ($1 \le M \le 100$), the number of holes. The next line contains a sequence of M positive real numbers that describe the diameters of the holes. The first such number is the diameter of hole 1, the second of hole 2, and so on.

A single number N ($1 \le N \le 26$), the number pegs, appears on the next line. N lines of input, each containing three real numbers describing the lengths of the sides of a triangular face, follow. The pegs are named 'A' through 'Z', so that first such line describes peg A, the second peg B, and so on.

Program Output

For each triangular peg, write one line indicating the holes that the peg will fit into, in the increasing numerical order given. Follow the format shown in the sample output. If a peg will not fit into any of the holes, output a line containing the text "Peg * will not fit into any holes", where * is the name of the peg.

Sample Input & Output

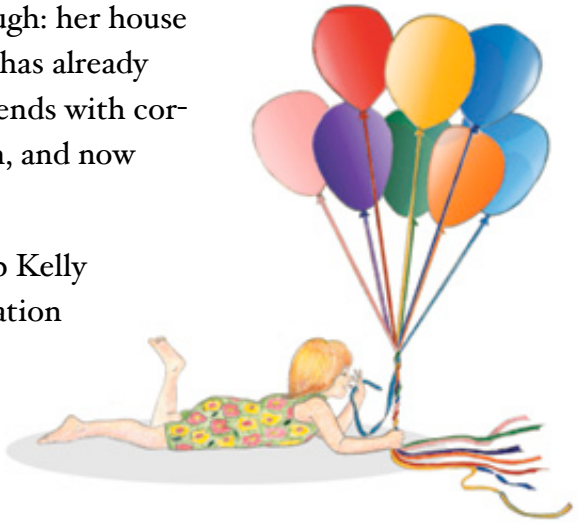
INPUT	OUTPUT
3 4.0 5.0 6.0 3 3.0 4.0 5.0 5.0 5.0 5.0 4.5 4.5 8.0	Peg A will fit into hole(s): 2 3 Peg B will fit into hole(s): 3 Peg C will not fit into any holes

Problem E

MIXING INVITATIONS

Input file: `invitations.in`
Output file: `invitations.out`
Java class name: `Invitations`

Kelly is having a birthday party soon, and would like to invite all her friends to come. There is one problem though: her house is far too small to accommodate everyone! She has already written up personalised invitations to all her friends with corresponding addressed envelopes to mail them in, and now she doesn't know what to do.



Luckily, Alex has thought of a clever way to help Kelly out. He knows that if a friend receives an invitation personalised to him or her, that friend will definitely come to the party. However, if a friend receives an invitation personalised to someone else (ie. the wrong invitation), that friend will surely not come to the party. Alex then goes to mix up Kelly's invitations and envelopes so that some invitations may be mailed to the wrong addressee. This way everyone still gets an invitation and nobody is left out, but no more people would come than would fit into Kelly's house.

There are exactly as many envelopes as there are invitations, and each invitation has a corresponding envelope with the same addressee. Alex must place each invitation into an envelope in such a way that there are no more invitations in their correct envelopes than the amount of people Kelly can accommodate.

Being the clever mathematician that he is, Alex would like to count the number of ways that he can mix up the invitations and envelopes to accomplish his goal. Can you write a program to do this for Alex?

Program Input

The input file contains several test cases, each on a separate line. Each line contains two positive integers, N and M, separated by a space. N ($1 \le N \le 12$) is the number of invitations Kelly has written, and M ($0 \le M \le N$) is the maximum number of people Kelly can accommodate.

Program Output

For each test case, output on a separate line a single integer: the number of ways Alex can mix up the invitations and envelopes to accomplish his goal.

Sample Input & Output

INPUT	OUTPUT
3 2 4 1 4 4	5 17 24

Problem F

PLAYING BOGGLE

Input file: `boggle.in`
Output file: `boggle.out`
Java class name: `Boggle`

Boggle® is a classic word game played on a 4 by 4 grid of letters. The letter grid is randomly generated by shaking 16 cubes labeled with a distribution of letters similar to that found in English words. Players try to find words hidden within the grid.

Words are formed from letters that adjoin horizontally, vertically, or diagonally. However, no letter may be used more than once within a single word.

An example Boggle letter grid, showing the formation of the words "taxes" and "rise".

The score awarded for a word depends on its length, with longer words being worth more points. Exact point values are shown in the table below. A word is only ever scored once, even if it appears multiple times in the grid.

No. of Letters	3	4	5	6	7	8 or more
Points	1	1	2	3	5	11

In this problem, your task is to write a program that plays Boggle. Given a letter grid and a dictionary of words, you are to calculate the total score of all the words in the dictionary that can be found in the grid.

Program Input

The first line of the input file contains a number N, the number of Boggle games that follow.

Each Boggle game begins with 16 capital letters arranged in a 4 by 4 grid, representing the board configuration for that game. A blank line always precedes the letter grid. Following the letter grid is a single number M ($1 \le M \le 100$), the number of words in your dictionary for that game. The next M lines contain the dictionary words, one per line, in no particular order. Each word consists of between 3 and 16 capital letters. No single word will appear in the dictionary more than once for a given Boggle game.

Program Output

For each Boggle game in the input, your program should output the total score for that game. Follow the format given in the sample output.

Sample Input & Output

INPUT	OUTPUT
2 TNXO AAEI IOSR BFRH 8 TAXES RISE ANNEX BOAT OATS FROSH HAT TRASH FNEI OBCN EERI VSIR 1 BEER	Score for Boggle game #1: 6 Score for Boggle game #2: 1

Problem G

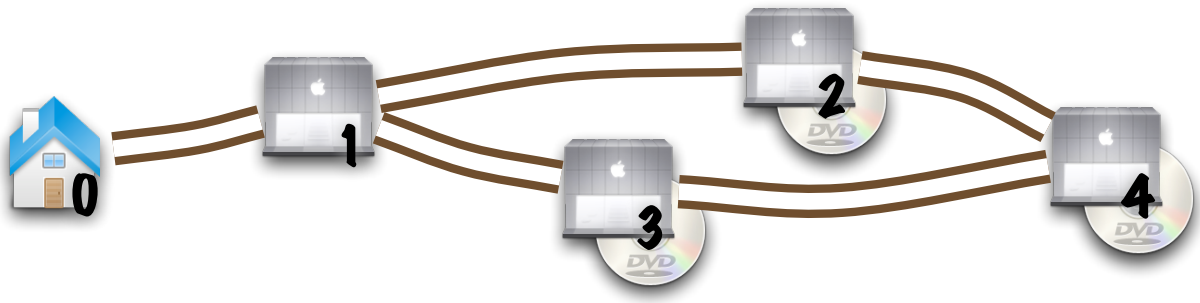
SHOPPING TRIP

Input file: `shopping.in`
Output file: `shopping.out`
Java class name: `Shopping`

For some reason, Daniel loves to collect and watch operas on DVD. He can find and order all the operas he wants from Amazon, and they will even deliver them right to his door, but he can usually find a better price at one of his favourite stores. However, with the cost of gas nowadays, it is hard to tell whether or not one would actually save money by driving to the stores to purchase the DVDs.

Daniel would like to buy some operas today. For each of the operas he wants, he knows exactly one store that is selling it for a lower cost than the Amazon price. He would like to know if it would actually be worth it to go out and buy the operas from the stores.

Daniel only knows the road system connecting his favourite stores, and will only use those roads to get around. He knows at least one route, if only an indirect one, to every store.



An example diagram of Daniel's house, his favourite stores, and the roads connecting them.

In his shopping trip, Daniel begins at his house, drives from store to store in any order to purchase his operas, then drives back to his house. For any particular opera, he can opt not to drive to the store to buy it, since he can just order it from Amazon.

For convenience, Daniel assigned his house the integer 0, and numbered each of his favourite stores with integers starting at 1. You are given a description of the road system and the exact amount it would cost for Daniel to drive each road. For each opera Daniel wants, you are given the number of the store it is available at, and the amount he would save if he bought that particular opera at that store. Your task is to determine the greatest amount of money Daniel can save by making the shopping trip.

Program Input

The first line of input contains a single number indicating the number of scenarios to process. A blank line precedes each scenario.

Each scenario begins with line containing two numbers: N ($1 \le N \le 50$), the number of stores, and M ($1 \le M \le 1000$), the number of roads. The following M lines each contain a description of a road. Each road is described by two integers indicating the house or stores it connects, and a real number with two decimal digits indicating the cost in dollars to drive that road. All roads are two-way.

The next line in the scenario contains a number P ($1 \le P \le 8$), the number of opera DVDs Daniel wants to buy. For each of the P operas, a line follows containing an integer indicating the store number at which the opera is available, and a real number with two decimal digits indicating the difference between the Amazon price and the price at that store in dollars.

Program Output

For each scenario in the input, write one line of output indicating the largest amount of money, in dollars and cents, that Daniel can save by making his shopping trip. Follow the format of the sample output; there should always be two digits after the decimal point to indicate the number of cents. If Daniel cannot save any money by going to the stores, output a single line saying "Don't leave the house".

Sample Input & Output

INPUT	OUTPUT
2 4 5 0 1 1.00 1 2 3.00 1 3 2.00 2 4 4.00 3 4 3.25 3 2 1.50 3 7.00 4 9.00 1 1 0 1 1.50 1 1 2.99	Daniel can save $3.50 Don't leave the house