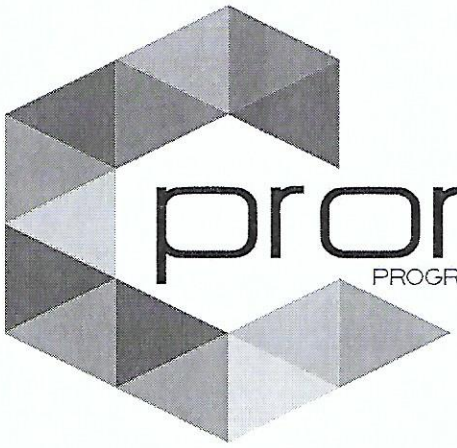


UNIVERSITI TEKNOLOGI MARA

UTP
UNIVERSITI TEKNOLOGI PETRONAS

prommcs
PROGRAMMING AND MULTIMEDIA COMPETITION 2017

PROGRAMMING COMPETITION
PROBLEM SET

Organized by:

Computer Science Department

Faculty of Computer and Mathematical Sciences

UiTM Perak Branch, Tapah Campus

Co-organizer:

Universiti Teknologi Petronas

UNIVERSITI TEKNOLOGI MARA

UTP
UNIVERSITI TEKNOLOGI PETRONAS

prommcs

PROGRAMMING AND MULTIMEDIA COMPETITION 2017

Problem Set

Instructions

Please read these directions carefully!

The following pages contain the problem set for the Programming Competition (ProMMCS'17) 2017. There are **TEN (10)** problems. You have **THREE (3) hours** to solve these problems. Some problems are easier than others. You will receive the same credit for solving the easy problems, as you will for solving the difficult ones, so it is in your best interest to read through all the problems before you begin solving.

IMPORTANT: For each problem you need to submit the source file. Note that the source code will not be evaluated. Your source file should be named **GroupName_problem*n*.cpp** (C++ program) or **GroupName_problem*n*.java** (Java program) where group name is a special name given to each group and ***n*** is the problem number.

The input and output for the problem is through standard input output. You need to create your own test data. The judges may use different set of test data. Pay careful attention to the output-formatting instructions in the problem descriptions.

Good Luck!

PROBLEM 1 : BALLOON COLLECTING

The UiTM Engineering Society has recently held a contest to develop the world's best balloon flotation compound. The team that won the competition has managed to develop a new gas compound that is able to float a conventional balloon well beyond that of helium gas. The team called their invention the "Super Hyper-Mega Global Anti-Gravitational Liquid Thermamine Compound", or the term SHMGALTC for short. Due to the excitement of winning the competition, the winning team released a large number of balloons into the air; however, these balloons are beginning to interfere with the airspace over Fredericton used for approaches to the airport.

UiTM has been ordered to collect as many of the balloons as possible, and they need your help. UiTM has only one large balloon capable of carrying human passengers, and they only have enough gas left to lift the balloon once, and then descend to the ground. All balloons will thus need to be caught as they descend. Thus UiTM's collecting balloon must lift to a certain height and then begin its descent, capturing the balloons one at a time as they descend. The balloons must be captured in order; they can skip a balloon in the sequence, but once skipped it cannot be captured later. They need your help to figure out the maximum number of balloons that can be caught.

The Input

The input contains one or more test cases, where each test case will consist of one or more non-negative integers representing the list of incoming balloons. These balloons must be processed in the order that they are given in. Each test case is terminated by a −1 (which does not represent one of the balloons), and the test cases are terminated by the end of file.

Note that there is no constraint on the number of balloons in the input.

The Output

The output should consist of one line per test case, stating "The maximum number of balloons possible to catch is: X", where X is replaced by the maximum number of balloons that you should be able to collect. Each test case should be on its own line.

Sample Input:

```
389 400 100 56 78 34 1 100 300 -1
1 2 3 3 2 1 -1
```

Sample Output:

```
The maximum number of balloons possible to catch is: 5
The maximum number of balloons possible to catch is: 3
```

PROBLEM 2 : CAN I HAVE RM1?

Rizqi is a pretty good guy — especially because it's REALLY easy to ask Rizqi for a 1 ringgit. If you ask Rizqi for a 1 ringgit multiple times, he always alternates between two responses:

"Uhh, I dunno..." which means he doesn't give you a 1 ringgit.
"Uhh, I guess..." which means he does give you a 1 ringgit.

For each person, he always starts with "Uhh, I dunno..." because he doesn't really like giving out money.

Given the number of times someone asks Rizqi for a 1 ringgit, calculate how much money Rizqi will give that person.

The Input:

The first line of the input will begin with a single, positive integer, *n*, representing the number of people asking Rizqi for money. Each situation will contain one integer on a new line, *x* ($0 \leq x \leq 10{,}000$), corresponding to how many times that person will ask Rizqi for a one ringgit.

The Output:

Output "`Person #i: RMp`" for each person *i* in the input (starting with 1) where *p* corresponds to the amount of money Rizqi gives that person.

Sample Input:

```
5
2
6
75
13
1
```

Sample Output:

```
Person #1: RM1
Person #2: RM3
Person #3: RM37
Person #4: RM6
Person #5: RM0
```

int testCase;
cin>>testCase
for(i=0; i<testCase; i++)
{
~~digit to test~~
cin >> inputNumber;
inputNumber = floor(inputNumber/2)
Cout << "Person #1: RM" << inputNumber << endl;
}

PROBLEM 3 : BABY NAMES

A new fad has arisen in the area of baby names. Parents no longer prefer to call their children standard, meaningless names. Rather, they now commonly combine both of their names to form one for the child which will be perfect and unique. The parents combine their names in the following way:

1. Both the expectant mother and father write their names out on paper.
2. Next, the parents decide whose name will be the beginning and whose will be the ending.
3. Then, they take the name for the beginning and cross out letters from the back of that name. At least one letter will be crossed out and at least one letter will be left. The same crossing out of letters is done for the name being used for the ending but from the front this time. However, now when crossing out the letters, if the beginning part ends in a vowel, the ending part must start with a consonant (Y is considered a consonant, not a vowel); similarly, if the beginning part ends in a consonant, the ending part must start with a vowel. Furthermore, neither the beginning nor the ending should be left with no letters.
4. Finally, the shortened ending is appended to the shortened beginning and a potential baby name has now been created.

An example of the parents doing this process can be seen below (note: this is just one of many ways the parents could combine their two names to form a baby name).

1. Write names out
   - Father's name: JAMES
   - Mother's name: MARY
2. Decide ordering
   - First half: JAMES
   - Latter half: MARY
3. Shorten the names
   - Shortened first half: JA
   - Shortened second half: Y
4. Combine the names
   - Baby name: JAY

Your task is to write a program which, given the names of the two parents, will output every possible valid name that can be formed from the process above. Furthermore, the parents want this list in alphabetical order with no repeats so they can easily sort through it.

The Input:

The first line of the input will contain a single, positive integer, n, representing the number of couples that are expecting a baby and need your help in creating the perfect name. The next n lines will each contain the Father's name followed by a single space and then the Mother's name. Each name consists entirely of capital letters from the standard English alphabet. Each of the parents' names will be at least 2 letters long and no longer than 20 letters.

The Output:

For each couple, first output "`Couple #i: p possible names`" where i represents the couple number from its order in the input (starting with 1) and p represents the number of different valid names that can be formed. Then, the next p lines should each contain one possible valid baby name consisting only of capital letters. All the valid names for one couple should be in alphabetical order with no repeats. It is, in theory, possible that one potential name for the baby is the same as one of the parent's names. This is okay, and the parents still want that name included in the list. Output an extra line after the output for each couple.

Sample Input:

```
2
JAMES MARY
ABE JO
```

Sample Output:

```
Couple #1: 10 possible names
JAMARY
JAMERY
JAMEY
JARY
JAY
MAMES
MARAMES
MARES
MAS
MES

Couple #2: 2 possible names
ABO
```

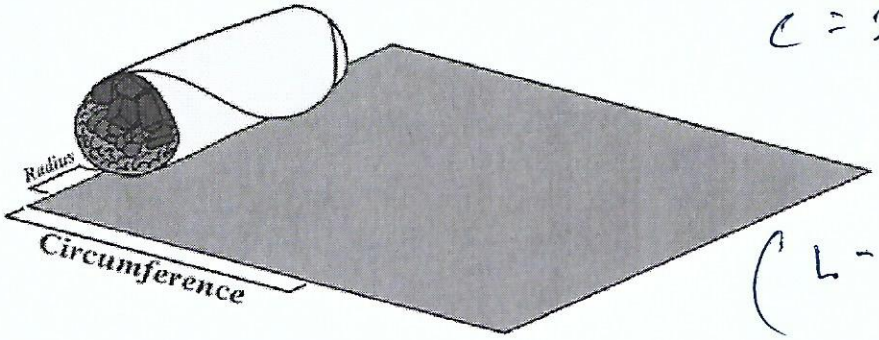
PROBLEM 4 : ROLLING KEBAB

Everyone knows that Ahmad loves circles, but sometimes he needs to stop admiring his circles and take a food break. Ahmad tends to eat the kebabs from the Paristanbul of Fast Food, as they are one of the most delicious circular foods on this planet. But sometimes his kebab doesn't fit in the foil wrappers, leading to a disappointingly cold kebab.

As everyone knows, kebabs are cylindrical which means they have a radius, length, circumference, and volume. A kebab can be completely wrapped if the rectangular foil is at least as long or as wide as the circumference, and the excess foil at each end of the kebab is at least its radius. The foil may not be able to wrap the kebab in its original orientation, so Ahmad will suggest that it can be rotated 90 degrees to ensure the kebab is warm.

$$volume = \pi * radius^2 * length$$

$$\pi = 3.141592653589793$$



Above is an example of a kebab that can be completely wrapped.

Given the volume and radius of a kebab, and the width and height of the foil, determine whether or not the kebab will fit and stay warm.

The Input:

The first line of the input is a single, positive integer, n , representing the number of kebabs in the input to check. Each kebab and foil is described in a pair of lines. The first line describes the kebab and will have a line containing two integers: the kebab's volume, v $(0 < v \leq 100)$, and its radius, r $(0 < r \leq 20)$, respectively. The second line for each kebab will denote the foil and will contain two integers: the foil's width, w $(0 < w \leq 50)$, and its length, l $(0 < l \leq 50)$, respectively. The kebab's volume is given in cubic inches where as the kebab's radius and the foil's dimensions are given in inches.

The Output:

For each kebab, first output " `Kebab # i :` " where *i* is the number of the kebab in the input (starting with 1), and then immediately follow this with either:

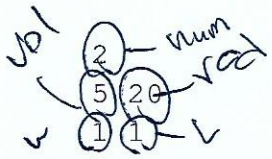
```
Don't worry, the kebab fits!
```

if the kebab will fit in the foil, or:

```
Looks like a cold kebab today.
```

if it will not.

Sample Input:



Sample Output:

```
Kebab #1: Looks like a cold kebab today.
Kebab #2: Don't worry, the kebab fits!
```

PROBLEM 5 : NEXT!

Rayyan has a very important appointment this morning, and has dutifully arrived very early for it. Unfortunately it seems as though everyone who has an appointment this morning also arrived early. Rayyan has been forced to take a number and wait his turn with everyone else in the waiting room. What is worse is that Rayyan can smell the sweet aroma of hash browns and coffee wafting from the nearby cafeteria! He really wants to enjoy an early morning coffee, but he also knows that if he is not in the waiting room when his number is called he will have to begin waiting all over again. He is not allowed to leave the cafeteria with his food or drink; the rules strictly prohibit such substances in the hallways or waiting room. Fortunately, he has some information to work with. He has seen several numbers called already, and so can estimate how long an appointment takes. He has had many a cup of coffee in his lifetime and so can estimate how long his cafeteria trip will take. He also knows his own number and the most recent number called. Write a program to advise Rayyan as to whether or not he has enough time to go get coffee or not.

The Input

The first line of the input contains a number $n > 0$ that represents the number of test cases that follow. n lines follow, each representing a test case. A test case consists of 4 integers a, c, i, j. a is the length of an appointment. c is the time it will take Rayyan to have his coffee (including travel to and from the cafeteria). i is the most recent number called for an appointment. j is Rayyan's number. All times are in minutes and all numbers are positive.

The Output

Each test case should result in a single line of output. If there is enough time for Rayyan to have his coffee before his number is called, output "Go for it!" Otherwise, output "Not enough time." You may assume the decision is made the very instant the number i is called, and that all numbers between i and j will be called in order. Rayyan will not miss his appointment if he gets back to the waiting room at the exact instant his number is called.

Sample Input:

```
4
5 10 103 106
5 16 15 18
20 19 99 100
15 5 1234 1234
```

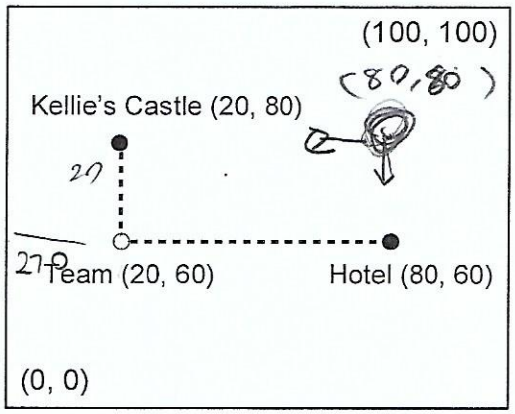
Sample Output:

```
Go for it!
Not enough time.
Go for it!
Not enough time.
```

PROBLEM 6 : LOST IN PERAK

The programming team was enjoying a day of sightseeing in Perak, but then Amir took the lead. Now, they are desperately lost, with only a few hours of daylight remaining to get back to their hotel. Fortunately, they are equipped with compasses and maps, and can deduce their location from the relative positions of certain landmarks. The team has a lot of practice being lost, and will always pick two landmarks that can be used to determine their location exactly.

For example, we can specify two landmarks on a grid: Kellie's Castle at grid location (20, 80) and the team's hotel at grid location (80, 60). Using this information, we can compute that the team is at location (20, 60). See the diagram on the right.



Given a list of landmarks and their locations from the map, and the compass headings of two of those landmarks, decide where the team must be lost in Perak.

The Input:

The first line of input will contain a single integer, m, ($2 \leq m \leq 100$) which is the number of landmarks on the map. The next m lines each contain an alphanumeric string, l, of 1 to 15 characters and two integers, x and y, each separated by a single space. The string l is the name of the landmark, and the two integers ($0 \leq x, y \leq 100$) represent its x- and y-coordinates on the map.

The next line contains a single positive integer, n, which indicates the number of data sets that follow. Each data set consists of two lines. Each line will contain an alphanumeric string s and a number h, separated by a single space. The string s is the name of the landmark, and will precisely match one of the names provided on the map. The number ($0.0 \leq h < 360.0$) is the compass heading of that landmark from the current position of the team. A value of 0.0 degrees for h represents due north, 90.0 degrees represents east, etc...

The Output:

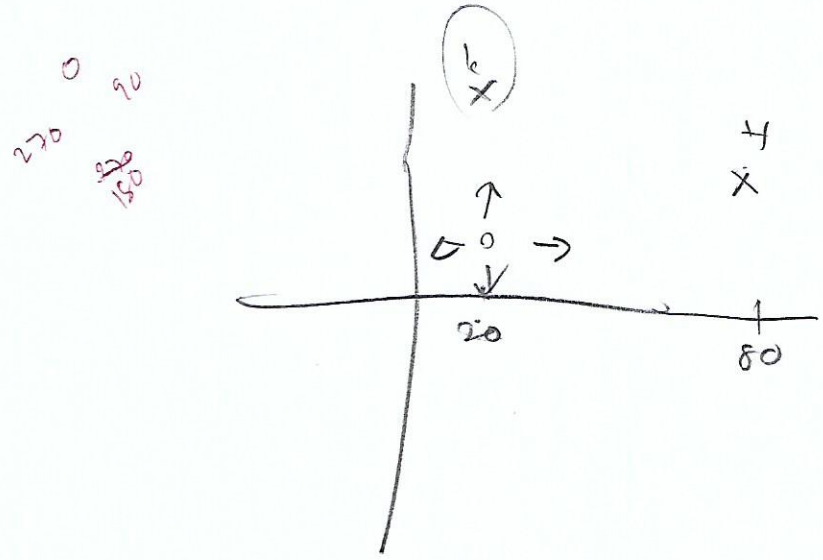
For each data set, print a line containing the message "`The team is lost at x, y!`" where x and y are the coordinates of the team on the map. Each of these values should be rounded to the nearest integer (0.5 rounds up).

Sample Input:

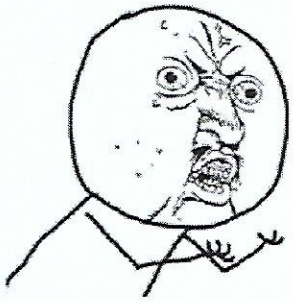
```
2
KelliesCastle 20 80
Hotel 80 60
2
KelliesCastle 0
Hotel 90
Hotel 180
KelliesCastle 270
```

Sample Output:

```
The team is lost at 20, 60!
```



PROBLEM 7 : A MEME GENERATOR



Generate "Y U NO" memes.

The Input:

The first line of input contains a positive integer, which is the number of memes to generate using the rest of the input. Data for each meme occupies 2 lines, starting on the next line. The first line is a string representing an *object of frustration*. The second line is a string containing the *expected action* for that object of frustration. Each line will contain at least one character and no more than 80 characters. There are no blank lines in the input.

The Output:

Output each meme on 2 lines as follows. On the first line, print the *object of frustration* followed by an ellipsis ("..."). On the second line, print "Y U NO" followed by the *expected action*, followed by a question mark. Capitalize all letters in the output. Print a blank line after the output for each meme.

Sample Input:

```
2
wi-fi
free at Starbucks
this program
compile on first try
```

Sample Output:

```
WI-FI...
Y U NO FREE AT STARBUCKS?

THIS PROGRAM...
Y U NO COMPILE ON FIRST TRY
```

PROBLEM 8 : THE MIRROR OF DECEPTION

Oh no! It's happened again! The evil dwarf magician, Gimlidwarf, has taken the princess Arwen captive. This time, he's brought together pieces of an ancient broken mirror in attempt to unite the worlds of light and dark. Once merged, he plans to bring treachery and deceit to Lothlorien, the kingdom of light. Like the epics passed through the generations, the land of Lothlorien is in need of its green-clothed hero, Glorfindel. But this time, Glorfindel needs your help!

It's been almost a year now since the announcement of the next title in The Epic of Rohan series of hit games. As an anxious gamer, you have your copy preordered for your GameSquare home entertainment system. Unfortunately, the GameSquare has been unofficially discontinued and will soon be replaced by the over-hyped and relatively rare Woo system. You may replace your preorder with a version for the Woo, so you decide to do some research about the game.

In The Epic of Rohan titles, Glorfindel has an unfortunate habit of being chosen to save Lothlorien from the deceitful Gimlidwarf. To do so, Glorfindel must slash his way through many elaborate dungeons and defeat the boss in each one. This will weaken Gimlidwarf. When all of Gimlidwarf 's minions are defeated, Glorfindel has a chance at a man-to-dwarf fight, giving Lothlorien the upper hand.

Each boss can only be threatened by a special weapon hidden deep within its own lair. It would be easy for Glorfindel if the weapon was readily available, but that wouldn't be very... epic... now would it? While the dungeons seem complicated at first, Glorfindel has noticed (through decades of unwanted experience) that the dungeons are much easier to navigate when given a descriptive map. Better yet, he can always describe the maps with an ASCII block of text. Each map indicates the locations of several locked doors, mini-bosses, treasures, and other surprises.

You have found out that the only difference between the GameSquare and Woo version of the newest installment is how Glorfindel uses the dungeon maps. Unfortunately, Glorfindel has a problem understanding the maps between the two versions of the game, and cannot navigate through any dungeon to the boss. Glorfindel loves to go through dungeons as fast as possible, but he needs your help to do this in the new version. You must write the program that helps him.

While ASCII maps have aided Glorfindel in his past epics, some newly added confusion complicates things. To help you help him, he has provided an ancient tablet. Unknown to him, it is a description for each of the map's markings.

Symbols:

```
B: Boss
C: Treasure Chest
D: Unlocked Door
E: Entrance
G: Giant Key
L: Locked Door
M: Mini-Boss
S: Small Key
W: Weapon
*: Wall
.:Floor
```

Example Dungeon Map:

```
*********B******
**....*...*W.*S*
*S.**...*.**M*.*
*********.**L*.*
*.............*
*L******L*****.*
*....S*.....G*C*
*E**************
```

Given this information, you must process each dungeon map to help Glorfindel find his way out of the dungeon. Glorfindel entered at the Entrance (`E`) and can walk through Floor (`.`) squares, but not through Wall (`*`) squares. The only way out is to find the Weapon (`W`) that kills the boss and the Giant Key (`G`) that opens the Boss (`B`) room. Once Glorfindel enters a Boss room with the correct weapon, consider it an easy victory. There is often a Miniboss (`M`) that Glorfindel may choose to defeat (which may or may not be required in order to exit the dungeon). Unfortunately, Glorfindel may also need to find several Small Keys (`S`) to unlock the Locked doors (`L`) along the way. Once a door is unlocked, the key breaks. Therefore, Glorfindel must find a small key for each locked door (note that he must find the key in an accessible area). Unlocked doors (`D`) require no key and may be used at any time. There are also Treasure Chests (`C`), but Glorfindel does not need to open these, as he always seems to have a full wallet.

The game designers have done a very good job testing the GameSquare version of the game. In each dungeon, Glorfindel will be able to find the weapon before he enters the Boss room. Also, in order for Glorfindel to encounter one of Gimlidwarf's minions, he will have to use all of the available small keys. You may assume that the area surrounding the map is inaccessible. Of course, there is always at least one route from the entrance to the boss room.

After investing some additional time researching the games, you have finally determined why Glorfindel is having trouble navigating through the dungeons. When looking at a particular map, the left side in the GameSquare version has become the right side in the Woo version, and vice versa. Luckily for you, Glorfindel still understands how to navigate through the GameSquare version. So, you must write a program to help Glorfindel through each dungeon of the Woo version by mirroring each map along the vertical axis.

The Input:

There will be multiple dungeons described in the input. Each dungeon starts with two integers, m and n ($5 \leq m, n \leq 80$), on a line by themselves separated by a single space. The next n lines will contain exactly m characters each. Only characters from the tablet will appear in the ASCII map descriptions. The last dungeon will be followed by a line containing two zeroes. This line should not be processed.

The Output:

For each dungeon, you must first output the header "`Dungeon x:`" where x is the dungeon number, starting from 1. Then, output the corresponding mirrored dungeon map. That is, each row of the map must be reversed so Glorfindel may find his way. After each map, output a blank line.

Sample Input:

```
5 5
.D..E
W*...
****.
.....
G.*B*
16 8
********B******
**....*...*W.*S*
*S.**...*.**M*.*
********.**L*.*
*.............*
*L******L*****.*
*....S*.....G*C*
*E**************
0 0
```

Sample Output:

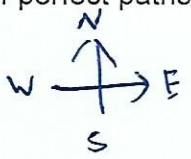
```
Dungeon 1:
E..D.
...*W
.****
.....
*B*.G

Dungeon 2:
******B*********
*S*.W*...*....**
*.*M**.*...**.S*
*.*L**.*********
*.............*
*.*****L******L*
*C*G.....*S....*
**************E*
```

PROBLEM 9 : HOW MANY PERFECT WAYS TO WORK?

Sara enjoys the beauty and perfection in nature. Unfortunately, Sara lives in a huge city that is mostly devoid of the trees and serenity of nature. Instead, Sara's city has many, many roads set up in a perfect grid, so she must settle for finding perfect paths within this grid. In the grid, a path consists of a sequence of movements in one of four possible directions: north (N), south (S), east (E) or west (W). Assume that each movement in a direction is a single block. A perfect path does not contain any movements in directly opposite directions. Thus, the sequence of movements NNENE forms a perfect path, but the sequence NEESN does not, since it contains both a north and south movement. To complicate matters, there are some street intersections that have graffiti, which Sara considers imperfect. Thus, any path that goes through these intersections is not a perfect path.

Given grid coordinates for Sara 's starting location and destination, as well as coordinates of all the imperfect intersections Sara is to avoid, you are to determine the number of perfect paths Sara can take.



The Input:

There will be several sets of input. The first line will contain a single positive integer n $(n < 100)$ describing the number of test cases in the data set. The first line in each data set has a single integer m $(0 \leq m < 10)$, which represents the number of intersections in the town for that particular data set that Sara must avoid. The following m lines contain the coordinates (x, y) of the m intersections to avoid. All x and y coordinates will be non-negative integers less than 10, with the x coordinate appearing first on a line, followed by the y coordinate, separated by a single space. The next line in the data set will be a single positive integer p $(0 < p < 10)$ that represents the number of trips for which you will be calculating the number of perfect paths for that data set. The last p lines of the data set contain the p trips. Each line will contain two pairs of (x,y) coordinates. The first pair will be the coordinates for Sara 's starting location and the second pair will be the coordinates for Sara's destination. Each coordinate on each line is separated by a single space from the previous and subsequent coordinates. You are guaranteed that none of Sara 's starting locations or destinations will be an intersection she is supposed to avoid. All of these coordinates will also be non-negative integers less than 10 and be separated by spaces on each line.

The Output:

For each data set, you will output a single line header of the following format:

```
Data Set k:
```

where k is an integer in between 1 and n, inclusive.

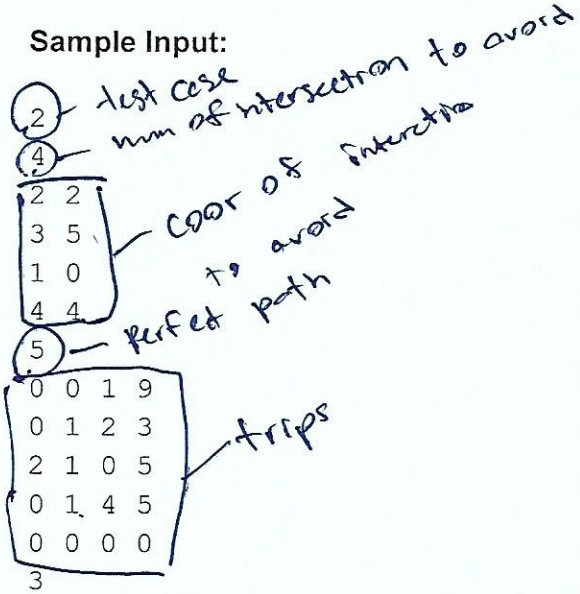
Follow this with a blank line, and then p lines, each with one of the two following formats.

```
Test Case c: Sara can take P perfect paths.

Test Case c: Sara can take 1 perfect path.
```

where *c* is an integer in between 1 and *p*, inclusive and `P` will be a non-negative integer less than 2147483647. Use the second format only if `P`=1. Please indent each of these lines exactly 2 spaces from the left margin. Also, leave a blank line in between data sets.

Sample Input:




```
2
4
2 2
3 5
1 0
4 4
5
0 0 1 9
0 1 2 3
2 1 0 5
0 1 4 5
0 0 0 0
3
1 1
2 2
3 3
1
4 4 9 7
```

Sample Output:

```
Data Set 1:

  Test Case 1: Sara can take 9 perfect paths.
  Test Case 2: Sara can take 3 perfect paths.
  Test Case 3: Sara can take 5 perfect paths.
  Test Case 4: Sara can take 0 perfect paths.
  Test Case 5: Sara can take 1 perfect path.

Data Set 2:

  Test Case 1: Sara can take 56 perfect paths.
```

PROBLEM 10 : KART RACING

Candy-coated kart racer, Vanellope Von Schweetz and friends are about to have their annual kart race. The winner of the race will win the prized candy-shaped trophy. However, Coily, a snake-like creature has decided he'd be better off making a fortune in coins by betting on the races instead of trying to win for himself. In the middle of the night, the evil snake sneaks into the garage and takes each kart for a test drive, measuring the kart's speed (oddly, each kart travels with a constant speed, without ever slowing down or speeding up). While the evil snake was out, he noticed a certain candy Cy-Bug walking around the course eating giant cherry and then carelessly dropping the cherry slimes on the road.

The evil snake found that these cherry slimes could not be avoided by any kart, and that every slime made a kart slip for five seconds, effectively stopping it in place. Once a kart hits a cherry slime, the slime is knocked from the road and subsequent karts will not be affected by it. Note that karts can freely pass other karts at any time. It is guaranteed that every course will only have one winner (no ties), and also guaranteed that only one kart will reach a given cherry slime (two karts won't reach the same cherry slime at the same time). In addition, cherry slimes are very slippery so no two cherry slimes will ever be at the same location.

In order to decide whom to bet on, the evil snake has kidnapped you to write a program to determine who will win the race.

The Input:

The first line of the input will contain a single positive integer, y, representing the number of circuits that the evil snake wants tested. Each circuit will begin with a single positive integer, m ($m < 9$), on a line by itself. Each of the following m lines will contain the data for one racer: the name of the racer (a string containing only upper and lowercase characters and hyphens, from 1 to 30 characters long) and the speed of the racer (an integer between 1 and 30 meters/second, inclusive), each separated by a single space from each other. Luckily, no racers have the same name within a single circuit so each name will appear in the list once.

Following the m lines, there will be a line with a single integer, n ($0 < n < 11$), representing the number of courses on the race circuit. On each of the following n lines there will be one race course description: the name of the course (a string containing only upper and lowercase characters and hyphens, 1 to 30 characters long), the length of the course in meters (an integer between 1 and 1000, inclusive), an integer b ($0 \leq b < 11$) representing the number of cherry slimes on the course, and finally b integers (each one between 1 and the course length, inclusive) denoting the distance from the track's starting point to a cherry slime. Within a circuit, each course has a unique name. Each component of the course description will be separated by a single space.

The Output:

For each circuit, first output, on a line by itself, a header "`Circuit #x:`" where *x* represents the circuit number (beginning with 1). For each of the courses within the circuit, list the name of the course followed by a colon and one space, and then the name of the racer who first travels the entire length of the course followed by one space and the phrase "`is the winner!`". Leave a blank line after the output for each circuit.

Sample Input:

```
2
3
Turbo 9
Vanellope 10
Taffyta 5
2
SugarRush-Track 60 3 10 14 59
Rainbow-Track 600 4 1 184 397 436
2
Vanellope 30
Candlehead 4
1
Jubileena-Track 200 2 194 14
```

Sample Output:

```
Circuit #1:
SugarRush-Track: Turbo is the winner!
Rainbow-Track: Vanellope is the winner!

Circuit #2:
Jubileena-Track: Vanellope is the winner!
```