Important Contest Instructions!!

***Please** read the following instructions carefully. They contain important information on how to run your programs and submit your solutions to the judges. If you have any questions regarding these instructions, please ask a volunteer before the start of the competition.*

Program Input
Most programs will require input. You have two options:

1) Your program may read the input from a file. The input data will be in the local directory in the file **probXX.txt**, where 'XX' is the problem number.

2) Your program may read the input from the keyboard (standard in). You may type everything on the keyboard, or you may copy the data from **probXX.txt** into the standard in. ***Tip:** Type 'Ctrl-Z <return>' to signal the end of keyboard input.*

   Note: An easy way to enter keyboard data is by redirecting the contents of a file to your program. For example, if you are executing prob01, the input file **prob01.txt** can be redirected to the standard in of your program using syntax like this (examples are shown for each of the allowed languages):

```
%> java prob01 < prob01.txt
%> java –jar js.jar prob01.js < prob01.txt
%> python prob01.py3 < prob01.txt
%> prob01.exe < prob01.txt
```

   Your program will behave exactly as if you were typing the input at the keyboard.

Program Output
All programs must send their output to the screen (standard out, the default for any print statement).

Submitting your Programs

Interpreted Programs (Java, JavaScript, Python) Your program must be named probXX.java / probXX.js / probXX.py2 / probXX.py3, where 'XX' corresponds to the problem number. For Python, use the extension that matches the Python version you are using. Please submit only the source (.java, .js, .py2 or .py3). For java, the main class must be named probXX. Note there is no capitalization. All main and supporting classes should be in the default (or anonymous) package.

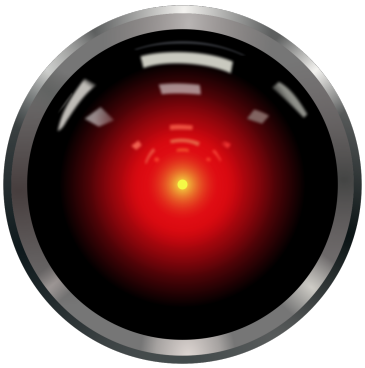
Native Programs (C, C++, etc.) Your program should be named probXX.exe, where 'XX' corresponds to the problem number.

You are <u>strongly</u> encouraged to submit solutions for Problems #0 and #1 (see next pages) <u>prior</u> to the start of the competition to ensure that your build environment is compatible with the judges' and that you understand the Input and Output methods

Problem.0 Hello Dave

+ 1 points

Summary



The sole purpose of this problem is to allow each team to submit a test program to ensure the programs generated by their computer can be judged by our judging system. Your task for this program is a variation on the classic "Hello World!" program.

In tribute to the 50th anniversary of the release of *2001: A Space Odyssey*, all you have to do is print "Hello, Dave. You're looking well today." to the screen.

Output

```
Hello, Dave. You're looking well today.
```

Problem.1 Introductions

+ 1 points

Summary

Without good communication skills, life will be more challenging for you. And if you don't know how to do Input and Output properly, the CodeWars competition will not go well for you. So, here's your chance for a little practice before the actual contest begins.

Write a program to introduce your team to someone by their first name.

Input

Input will consist of a single line with a single word that is your new friend's first name.

```
Wilbur
```

Output

Greet your new friend with a friendly, creative greeting of some sort that includes the person's name. Output should contain only the greeting and not any prompt for entering input.

```
Salutations, Wilbur! We are the Fighting Sandcrabs from Port Lavaca HS!
```

Problem.2 How Far Will You Go?

+ 3 points

Summary

In winter, the fjords of Erindale freeze over and the contestants prepare for the annual ice rocket contest. The goal of the contest is to launch a two-stage model rocket across the ice so that it stops as close as possible to the goal line without touching or passing over it. The winner gets to ride a reindeer in the winter parade.

Every year the exact distance from the starting line to the goal is kept a secret until a week before the contest. To win, contestants must do some math to determine the optimal thrust for their rockets. One of the equations they use is a simple distance calculation:

```
d = vt + (1/2)(at^2)
```

Using this formula, a contestant can determine the distance *d* a rocket will travel for a number of seconds *t*, given an initial velocity *v* and acceleration *a*.

Input

Each line of input contains three real numbers: a velocity *v*, acceleration *a*, and time *t*. The input ends with a line of three zeroes. The range of all input values is -100 to 100.

```
14.46 -1.38 8.7
0 11.34 2.4
63.5 2.1 4.4
97.01 0.00 1.313
0 0 0
```

Output

For each line of input, the program must print the distance traveled. Answers must be accurate to within +/- 1.

```
73.576
32.659
299.728
127.374
```

Problem.3 Magnanimous Numbers

+ 4 points

Summary

A magnanimous number is a number such that any insertion of a + sign between any two digits in base 10 results in an expression of a prime integer. A number that is not magnanimous is petty.

For example 40427 is magnanimous because

```
4+0427  = 431  is prime
40+427  = 467  is prime
404+27  = 431  is prime
4042+7  = 4049 is prime
```

Input

Each line of input is a positive integer with two to seven digits to be tested. The input ends with the number zero.

```
40427
819
101
109
2000221
4063
10
98
0
```

Output

For each input number, the program must print that number and the word MAGNANIMOUS or PETTY.

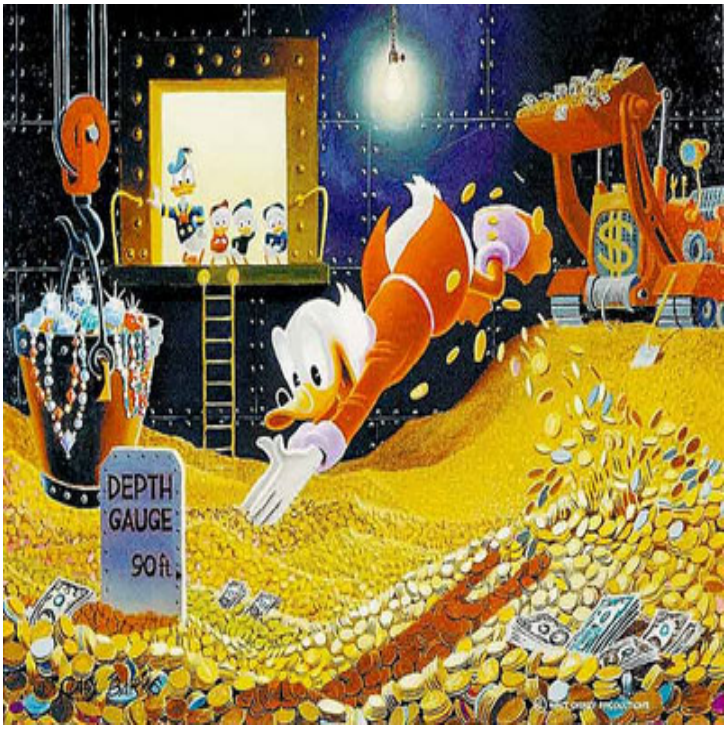
```
40427 MAGNANIMOUS
819 PETTY
101 MAGNANIMOUS
109 PETTY
2000221 MAGNANIMOUS
4063 MAGNANIMOUS
10 PETTY
98 MAGNANIMOUS
```

Problem.4 Make Room in the Money Bin

+ 4 points

Summary

Scrooge McDuck's vault is practically overflowing.



Normally, Scrooge wouldn't consider something like this a problem. But a recently passed city law in Duckburg mandates a minimum height for diving boards, and Scrooge's board is too close to the surface of his fortune.

The city will remove any boards that don't comply with the new rule. One of Scrooge's favorite activities is diving into his tower of treasures, so he wants to make sure his board doesn't get taken away.

He plans to make room in the vault by exchanging some of his coins. Scrooge has three types of coins in the vault: gold, silver and bronze. One gold coin is worth 10 silver coins and 50 bronze coins. One silver coin is worth five bronze coins.

Scrooge needs to make a lot of room in his vault, so he wants to end up with the fewest total number of coins that he can. But Scrooge also insists on keeping at least one of each type of coin in his vault after the exchanges are complete.

Given the number of gold, silver and bronze coins that Scrooge has in his vault, tell him the number of each type of coin he will have after the exchanges.

Input

Input begins with a single integer C (0 < C < 20) representing the number of test cases to follow.

The following C lines will contain three integers G, S and B (0 < G, S, B < 10000) representing the number of gold, silver and bronze coins in Scrooge's vault.

```
3
500 400 300
5 5 5
59 79 99
```

Output

For each test case, output the number of gold, silver and bronze coins Scrooge will have after the exchanges.

```
545 9 5
5 5 5
68 8 4
```

Problem.5 Mark Your Calendars

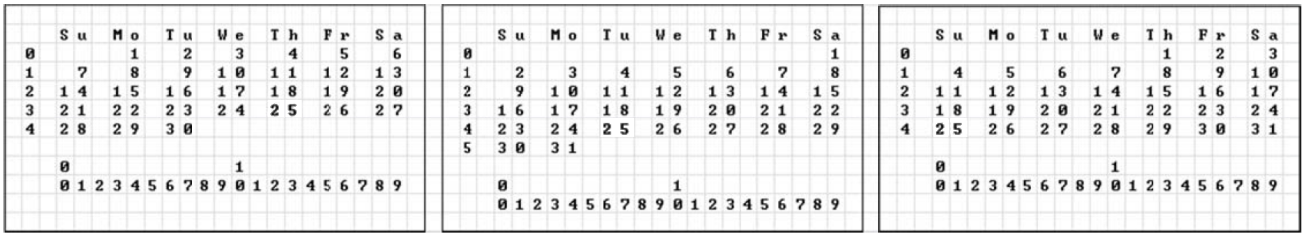
+ 4 points

Summary

The CodeWars contest day is an anticipated event every year, and you want to make sure you don't miss it.

You have a new desktop LED calendar that can be programmed to light up on specified days. The calendar consists of 6 rows and 20 columns of LEDs for displaying the dates for each month. Each digit of a date takes up a single LED square on the calendar, and there is a single square between each day of the week. For dates with a single digit, the digit will be in the rightmost square.

Three sample calendar layouts are pictured below, with reference row and column indexes to the right and below, respectively.



Given the day of the week that a month begins and the date of the CodeWars contest, output the row and column(s) that should be lit to highlight that date.

Input

Input begins with a single integer C (0 < C <= 20) representing the number of test cases to follow.

The following C lines will contain a single string followed by a single integer. The string represents the day of the week of the first day of a month. The integer D (0 < D <= 31) represents the date of a holiday.

```
4
Monday 8
Monday 17
Thursday 1
Thursday 24
```

Output

For each test case, output a space-separated list of the row and column(s) that should be lit. The columns should follow the row in the output and be listed in ascending order.

```
1 4
2 9 10
0 13
3 18 19
```

Problem.6 Phone Number Validator

+ 5 points

Summary

Web sites often attempt to validate user input, albeit to varying degrees of success. Anecdotal experience indicates that most web sites collecting phone numbers for North American customers use validation criteria that are far too strict. Poor validation code leads users to frustration, confusion, and delay. Your gentle CodeWars team feels an obligation to future generations by preparing programming students with guidance for how easy it is to validate phone numbers without exasperating your users.

The North American Numbering Plan (NANP) is a telephone numbering plan that encompasses 25 distinct regions in twenty countries primarily in North America, including the United States, Canada, and the Caribbean. A valid NANP phone number has ten numeric digits. The first three digits are the Area Code. The next three digits are the Exchange Code. The final four digits are the Subscriber Number. The following rules apply to each component:

- The first digit of the Area Code may be 2-9 (0 and 1 are not permitted). The other digits may be any value 0-9, except that the NANP is not currently assigning area codes with 9 as the second digit.
- The first digit of the Exchange Code may be 2-9 (0 and 1 are not permitted). The other digits may be any value 0-9, except that the second and third digits cannot both be 1.
- The Subscriber Number may use any value 0-9 for all four digits.

Write a program to validate whether or not a phone number conforms to the NANP.

Input

The first line of input indicates the number of input lines that the program must validate. Each line thereafter is to be treated as a user input for a phone number.

```
8
1234567890
(879) 867-5309
(494)852-8921
(281)302-1492
281.302.1492
281.311.1492
281.012.1492
7138675309
```

Output

For each input line, the program must echo the line to output and print whether the input is VALID or INVALID.

```
1234567890 INVALID
(879) 867-5309 VALID
(494)852-8921 INVALID
(281)302-1492 VALID
281.302.1492 VALID
281.311.1492 INVALID
281.012.1492 INVALID
7138675309 VALID
```

Problem.7 Maximum Product of Three

+ 5 points

Summary

Write a program that finds the maximum product of three numbers in a list of integers.

Input

The first integer on each line indicates how many numbers N are in a list. The rest of the line contains N integers. The end of input is indicated when N is zero. The maximum value of N is 999. The integers will be in the range -999 to 999, inclusive. Download the sample data from the contest server to get the complete input set.

```
5 5 -3 2 4 -1
7 999 2 0 834 3 1 697
4 2 1 0 -1
8 5 2 -3 4 -4 1 0 -2
15 89 -2 2 3 -4 87 1 -98 -6 -97 -1 88 -3 -99 -5
6 0 -1 -2 -3 -8 -9
8 -2 -5 -3 -7 -1 -4 -8 -6
999 -322 354 -967 -291 385 -936 . . .  -77 599 -722 -46 630 -691 -15 661 999
0
```

Output

For each line of input, the program must print the maximum product of three numbers from the list. The maximum allowed run time is TEN SECONDS.

```
40
580716702
0
60
863478
0
-6
994010994
```

Problem.8 Serialization

+ 5 points

Summary

For this problem we define a mathematical operation called *serialize*. The serialize operation concatenates consecutive integers into a single string, beginning from a non-negative *start value*. The operation continues concatenating consecutive integers as long as the string length is less than or equal to a *length value*. The result of the operation is the value of the last integer concatenated onto the string. The operation result is zero if the string length of the start value is greater than the length value.

Start	Length	Serial String	String Length	Result
0	5	01234	5	4
0	11	0123456789	10	9
3	9	345678910	9	10
7	18	78910111213141516	17	16
98	9	9899100	7	100
13	1		0	0

Input

Each line of input contains a start value and a length value. The maximum of either value will be 8192. The input ends with two zeroes.

```
0 5
0 11
3 9
7 18
98 9
13 1
8192 1024
256 8192
0 0
```

Output

For each input, the program must print the input values and the result of the serialization operation.

```
0 5 4
0 11 9
3 9 10
7 18 16
98 9 100
13 1 0
8192 1024 8447
256 8192 2489
```

Problem.9 Electron Heavy Numbers

+ 5 points

Summary

When writing software, it can be easy to forget all the layers of abstraction underlying the code we write. As twitter user @daisyowl put it:

> If you ever code something that "feels like a hack but it works," just remember that a CPU is literally a rock that we tricked into thinking. Not to oversimplify: first you have to flatten the rock and put lightning inside it.

One of those layers of abstraction is that data, including numbers, are represented as binary sequences of ones and zeroes. A little intuitive reasoning suggests that the ones must be heavier than the zeroes, right?

Right? Work with me, here.

If so, then we can categorize numbers as *heavy* if the number of ones is greater than the number of zeroes (excluding leading zeroes), *light* if the number of zeroes is greater, and *balanced* if the the number of ones and zeroes are equal.

```
Decimal   Binary             Category
-------   -----------------  --------
      5                 101  heavy
      8                1000  light
     10                1010  balanced
     17               10001  light
    316           100111100  heavy
    632          1001111000  balanced
```

Write a program that can determine if a decimal input number is heavy, light, or balanced.

Input

Each line of input contains a single positive integer, with up to nine decimal digits. The input ends with a zero. The program does not need to determine if the zero is a decimal zero or a binary zero.

```
5
8
10
17
316
987654321
65536
8675309
0
```

Output

For each input number, the program must print the decimal number and the correct categorization: light, balanced, or heavy.

```
5 HEAVY
8 LIGHT
10 BALANCED
17 LIGHT
316 HEAVY
632 BALANCED
987654321 HEAVY
65536 LIGHT
8675309 HEAVY
```

Problem.10 The Cover Up

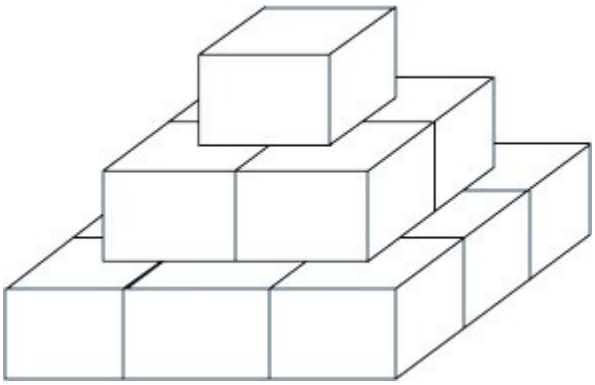
+ 6 points

Summary

The Sapphire Consulting and Marketing company is adding decorative pyramids to their corporate headquarters.

The company plans to build the pyramids using cubic blocks of stone with each side one meter long.

Sapphire has a precise method for building its pyramids: - The pyramid comprises a number of square layers, each with sides parallel to the base. - For each level, all sides of that level will have an equal number of blocks. - Each block must be centered at the intersection of four blocks in the level below. - A block will always be centered above the intersection of four blocks in the level below.



The company wants to make sure the pyramids reflect its image of wealth, and as such will be painting them all gold.

Six liters of paint are needed to adequately cover a single cubic block, covering every face of the cube equally.

But Sapphire didn't get where it is by wasting money and wants to use the minimum amount of paint. Only the parts of the pyramid that are visible when the pyramid is installed in Sapphire's campus will be painted.

Given the number of stones on the top level of the pyramid and its height, tell how many liters of paint are need to paint the pyramid.

Input

Input begins with a single integer C (0 < C <= 10) representing the number of test cases to follow. The following C lines will each contain two integers.

The first integer T (0 < T <= 250000) represents the number of stones on the top level of the pyramid. T is guaranteed to be a perfect square. The second integer H (0 < H <= 500) represents the height of the pyramid.

```
4
1 1
1 2
4 2
289 3
```

Output

For each test case, print the liters of paint needed for a pyramid with a top of T stones and height H. Output from each test case should be on a separate line.

```
5 liters
16 liters
29 liters
577 liters
```

Problem.11 Picking Teams

+ 7 points

Summary

Trillian and Slartibartfast are team captains in a match of Brockian Ultra-Cricket and are preparing to pick teams.

Team selection is done by a series of turns where captains choose zero or more consecutive players from a list. Selection continues until all players from the list are selected or both team captains pick zero players in consecutive turns.

The list contains each player's skill level and name.

Trillian won the coin toss, so she will pick first. Trillian wants her team to be as skillful as possible. She also wants as many players as possible, so she'll pick a group containing players with negative skill levels as long as the overall group has the greatest non-negative cumulative skill level available.

If multiple available groups of consecutive players have the same cumulative skill value, Trillian will take the group that comes first in the list.

Trillian will not make any selection that has a negative cumulative skill.

Given Trillian's constraints and a list of available players, tell who Trillian will select with her first pick.

Note: Solutions should complete all test cases in an input file within 10 seconds. Be sure to run against all sample student input files.

Input

Input begins with a single integer C (0 < C <= 10) representing the number of test cases to follow.

Each test case will begin with a single integer P (1 < P <= 200000) representing the number of players in the list.

The following P lines will describe a player. Each line begins with a single integer S (-100000 <= S <= 100000) representing a player's skill level. A single space separates the player's skill and the player's name, which will consist of at most 80 ASCII characters and can contain spaces.

Player names for each test case will be distinct.

```
4
5
10 Fenchurch
-14 Colin
2 Random Dent
5 Zaphod Beeblebrox the Fourth
3 Gag Halfrunt
2
-4 Prostetnic Vogon Jeltz
-1 Stavro Mueller
3
0 Zarniwoop
-1 Rob McKenna
0 Garkbit
5
6 Wonko the Sane
11 Arthur Dent
9 Ford Prefect
-37 Lig Lury, Jr
27 Zarquon
```

Output

For each test case, identify the players that Trillian selects with her first pick on a single line. If the pick has more than one player, output the names of the first and last players in the group separated by the word "to". If the pick has one player, output only that player. If there is no valid pick, output "None".

```
Random Dent to Gag Halfrunt
None
Zarniwoop
Zarquon
```

Problem.12 I'll Take the Case

+ 8 points

Summary

Harvey Birdman has been feeling pressure from his boss, Phil Ken Sebben, to bring in more money for the firm.

Phil has given Harvey a list of possible cases to take. But Harvey has a peculiar way of deciding whether he will take a case. Harvey is only interested in taking a new case if it is an anagram of one of his existing cases. (An anagram is the rearranging of letters from one word or phrase to form another word or phrase, using all of the original letters exactly once. The number of spaces can be different.)

You're given a list of Harvey's existing cases and Phil's list of cases for Harvey to take. For each case on Phil's list, write whether Harvey takes that case.

If Harvey takes a case, note which case among his existing cases is an anagram for the new case.

Input

Input will begin with an integer C (0 < C < 50) representing the number of cases Harvey currently has. The following C lines will each contain the name of one of Harvey's current cases.

Next will be an integer P (0 < P < 50) representing the number of cases for Harvey to consider. The following P lines will contain the name of a potential new case.

No case name will contain more than 100 characters. All case names will consist only of uppercase letters.

```
4
PETER POTAMUS V THAT THING I SENT YOU
PEOPLE VERSUS ROGERS AND DOO
DOE ONE VS PROLOG PERSAUDERS
PEOPLE V FLINTSTONE

5
POETS V FENNEL PILOT
KRAMER V KRAMER
PHOTOTYPESETTER V GIANT HUMAN SUIT
DOE VS PROSPEROUS GOLDEN ERA
PEOPLE V SEBBEN
```

Output

For each case that Harvey considers, write whether he will take the case.

If Harvey takes the case, write "Yes: " followed by the case from his current case list that's an anagram. If more than one case is an anagram, choose the one that comes first alphabetically.

If Harvey does not take the case, write "No: No matching case"

```
Yes: PEOPLE V FLINTSTONE
No: No matching case
Yes: PETER POTAMUS V THAT THING I SENT YOU
Yes: DOE ONE VS PROLOG PERSAUDERS
No: No matching case
```

Problem.13 We've Got You Covered?

+ 8 points

Summary

The CodeWars staff is always working to improve the student experience at our competitions, and this year we're experimenting with the placement of WiFi hotspots ... but we're not sure we've got it right.

The layout for each contest site is represented by an L by W grid with student locations marked with a # character.

WiFi hotspot locations are provided as X, Y coordinates that correspond with the length and width of the room grid, respectively. WiFi hotspots can be placed outside of the contest room. WiFi hotspots can also occupy the same location as a student.

Each WiFi hotspot has a an associated signal radius inside which reliable network coverage can be guaranteed. Students must be strictly inside a hotspot's radius to get reliable coverage.

Student and hotspot locations will be at the center of their grid square.

Given a set of contest sites, student locations, and WiFi placements, tell how many students will get reliable network coverage at each site.

Input

Input will begin with an integer S (0 < S <= 20) representing the number of sites.

The description for each site will begin with a single integer H (0 < H <= 20) representing the number of wireless hotspots at that site.

The following H lines will describe a single hotspot. A hotspot description begins with a pair of integers X and Y (-50 <= X, Y <= 250) representing its coordinates and ends with an integer R (0 < R <= 300) representing the radius of the hotspot's signal. The X coordinate corresponds with the length of the room and the Y coordinate with the width.

Next come a pair of integers L and W (2 <= L, W <= 200) representing the length and width of the room at a contest site. The first character of the first line is at coordinates 0, 0.

The following L lines will contain a string W characters long consisting of either a **.** or a **#**. A **.** represents an empty space and a **#** represents a student location.

```
2
3
0 1 2
3 3 1
5 -2 4
4 4
#.#.
 .##.
 .##.
#..#
2
1 4 1
2 2 1
3 5
.....
....#
.#...
```

Output

For each contest site, output a single line that tells how many students get coverage with that site's WiFi configuration.

For sites where a single student gets coverage, use the following format: "Site #: 1 student gets coverage"
For other sites, use the following format: "Site #: # students get coverage"

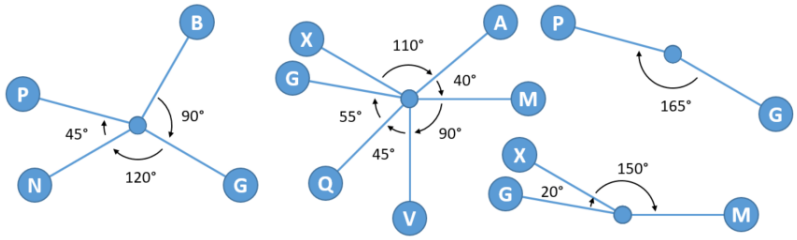
```
Site 1: 6 students get coverage
Site 2: 1 student gets coverage
```

Problem.14 Spoke Matching in Radial Systems

+ 9 points

Summary

Imagine a radial system consisting of a wheel with arbitrarily placed spokes. Each spoke is labelled with a unique upper-case letter, so two spokes in the same system cannot have the same label.



A radial system can be described as a list of integer angles and labels. The sum must be less than 360 degrees.

- `0 B 90 G 120 N 45 P`
- `0 X 110 A 40 M 90 V 45 Q 55 G`

Angles are always measured clockwise, but a system description can begin from any spoke. The following two descriptions are both valid for the same radial system:

- `0 B 90 G 120 N 45 P`
- `0 N 45 P 105 B 90 G`

A radial system can partially match a larger system if all the spokes of the smaller system line up radially with spokes that have the same label in the larger system. Spokes in the larger system may be skipped. For example:

- `0 G 165 P` partially matches `0 B 90 G 120 N 45 P`
- `0 X 110 A 40 M 90 V 45 Q 55 G` partially matches `0 G 20 X 150 M`

Write a program that can determine if two radial systems are an exact match, a partial match, or unmatched.

Input

The first line of input indicates the number of pairs of radial systems that follow. Each line after describes a pair of radial systems. Each radial system begins with a integer number of spokes (one to eleven, inclusive). Every spoke is described by an integer angle and a single upper-case letter.

```
10
4 0 B 90 G 120 N 45 P 4 0 N 45 P 105 B 90 G
3 0 G 20 X 150 M 6 0 X 110 A 40 M 90 V 45 Q 55 G
3 0 G 20 X 150 M 4 0 N 45 P 105 B 90 G
11 0 S 3 J 44 G 32 Z 75 U 9 Y 18 K 88 Q 16 X 25 B 39 E 9 0 K 88 Q 16 X 25 B 39 E 14 J 44 G 32 Z
8 0 X 80 V 23 R 39 T 31 X 60 K 44 H 56 D 8 0 K 44 H 56 D 26 X 80 V 23 R 39 T 31 X
2 0 X 180 Y 2 0 Y 180 X
5 0 F 100 W 70 N 45 Y 60 L 3 0 Y 125 F 170 N
7 0 E 2 P 44 S 32 H 75 L 29 A 67 X 3 0 S 107 L 96 X
7 0 E 2 P 44 S 32 H 75 L 29 A 67 X 3 0 S 107 A 96 X
8 0 T 31 X 60 K 44 H 56 D 27 X 80 V 23 R 8 0 X 80 V 23 R 39 T 31 X 60 K 44 H 56 D
```

Output

For each pair of radial systems, the program must print whether the pair is an exact match, a partial match, or mismatched.

```
EXACT
PARTIAL
MISMATCH
PARTIAL
MISMATCH
EXACT
MISMATCH
PARTIAL
MISMATCH
EXACT
```

Problem.15 Hexagon Word Search

+ 10 points

Summary

Write a program to solve a word search puzzle shaped as a hexagonal grid. Words may be hidden forward, backward, or any of the four diagonal directions.

Input

The first line of input indicates the number of search words and the size of the puzzle (i.e., the number of letters per side of the hexagon). The maximum of either value is 11. The search words come next, followed by the puzzle itself with the rows and columns of letters arranged to form a hexagon shape. No search word will be an exact substring of another search word.

```
5 6
PROGRAM
COMPILER
ERROR
SYNTAX
OBJECT
     L I P M O C
    E R P O R Y A
   X A O B J A C T
  R E L I P M O C N
 S I E R R O R E T Y
C Y P M O R Y J X O S
 S N Y G O G B L E R
  U T R R C O M P U
   N A R E R R O W
    M X G I C P U
     L S J N X A
```

Output

The program must locate the search words in the puzzle and print a solution to the puzzle. The printed solution must show the search words in their position in the puzzle and replace all other letters (i.e., not the search words) with periods.

```
     . . . . . .
    . . . . . . .
   . . . . . . . T
  R E L I P M O C .
 S . E R R O R E . .
. Y . . O . . J . . .
 . N . G . . B . . .
  . T R . . O . . .
   . A . . . . . .
    M X . . . . .
     . . . . . .
```

Problem.16 10-Digit Snacks

+ 10 points

Summary

The "10-Digits Everywhere" company has been doing well and now allows employees to access the new fully stocked snack room. However, anyone entering must prove they're worthy by solving a 10-digit problem to open the lock. Each month, the problem is different. Here is this month's:

The number displayed on this lock is the sum of two numbers. Taken together, the digits 0-9 appear exactly once in all the digits of the two numbers. Also, the product of the two numbers is a 10-digit number which also uses the digits 0-9 exactly once.

Will you be able to get snacks this month? Write a program which reads a single integer between 5 and 9-digits long representing the sum of 2 positive integers A and B. Find A and B and their product. Remember, the digits 0-9 appear once each in all the digits of A and B taken together, and once each in their product.

Input

The input is an integer N from 1 to 10.
The next N lines each contain a single integer sum between 10,000 and 999,999,999.

```
Example 1:
  1
  99999

Example 2:
  5
  123345
  181539
  181638
  7620732
  910473291
```

Output

For each sum, your program must print values for "sum : A * B = product". Let A < B.

```
Example 1:
  99999 : 47931 * 52068 = 2495671308

Example 2:
  123345 : 54237 * 69108 = 3748210596
  181539 : 9486 * 172053 = 1632094758
  181638 : 84015 * 97623 = 8201796345
  7620732 : 534 * 7620198 = 4069185732
  910473291 : 6 * 910473285 = 5462839710
```

Problem.17 Vactrain Packing

+ 11 points

Summary

Up-and-coming tech giant Abasin recently opened a central warehouse facility from which it ships packing crates by hyperloop to its regional distribution centers. Abasin has implemented an optimal packing policy in order to reduce shipping costs. Here's how the new process will work:

The loading dock has one packing platform dedicated to each of its remote distribution centers. Each platform has docking space for loading three vactrains, labelled A, B, and C. The packing crates are various sizes and arrive at the platform in sequential order from the warehouse queue. The platform supervisor is responsible for packing each vactrain as full as possible. This means that sometimes the dockbots will be directed to transfer a crate from the vactrain where it was loaded to another vactrain. Only the most recently loaded crate can be removed from a vactrain. Also, crates cannot be kept -- even temporarily -- on the packing platform. When a crate is removed from the warehouse queue, it must be immediately loaded into a vactrain. Likewise, if a crate is removed from a vactrain it must be immediately packed into another vactrain. Each vactrain has a crate size capacity of ten. When the supervisor is satisfied that a vactrain cannot be loaded further, it is sent to its destination and an empty vactrain takes its place at the platform.

Abasin needs to maintain correct manifests of each vactrain. Write a program to track the packing of crates and sending of vactrains so that it can print accurate manifests.

Input

Each line of input describes a single transaction. The first word indicates the transaction type: RECV for receiving a packing crate on the warehouse queue, LOAD for loading a vactrain from the warehouse queue, XFER for transferring a crate from one vactrain to another, and SEND for sending a vactrain to its destination. An input line for a receiving transaction includes a unique crate label and an integer crate size. An input line for a load transaction also includes a letter indicating the vactrain into which the crate is loaded. An input line for a transfer transaction includes two letters indicating the vactrain from which the crate was removed and the other vactrain into which the crate was transferred, in that order. A shipping transaction is followed by a single letter indicating which vactrain is sent to its destination. The input ends with the word DONE.

```
RECV LightningDust 3
RECV SmartFlatRocks 3
LOAD C
RECV ApplePies 3
LOAD B
RECV VerySmallRocks 2
LOAD A
RECV CommandBlocks 2
LOAD C
RECV StickyPistons 4
RECV LargeHadrons 3
LOAD A
XFER B C
LOAD B
XFER A C
SEND C
LOAD B
XFER A B
SEND B
DONE
```

Output

The program must print an accurate manifest of each vactrain, in the order which the vactrains were sent. The manifest must list the contents of the vactrain in removal order, which is reverse of the order in which it was loaded.

```
VACTRAIN 1
CommandBlocks 2
SmartFlatRocks 3
VerySmallRocks 2
LightningDust 3
VACTRAIN 2
ApplePies 3
LargeHadrons 3
StickyPistons 4
```

Problem.18 Word Connect

+ 12 points

Summary

A word connect is a puzzle in which a list of dictionary words must be fitted into a list of puzzle blanks. Each puzzle blank is encoded as a two-digit number that represents a letter. Every two-digit number represents the same letter throughout the puzzle. However, it is possible for the same letter to be represented by multiple two-digit numbers. There will be a set of dictionary words that can be applied to the word blanks to solve the puzzle. For example, consider the following dictionary and puzzle blanks:

```
Dictionary                    Puzzle
   APE                      ____ ____ ____
   EAT                       11   34   72
   EAR                      ____ ____ ____
   PIE                       34   88   72
   POT                      ____ ____ ____
   RAT                       72   11   64
```

The solution to this puzzle is APE, PIE, EAR.

Write a program that can solve a word connect puzzle.

Input

The first line provides the number of dictionary words (max 32) and the number of puzzle blanks (max 8). Next comes the dictionary with one word per line. Last comes the puzzle blanks, one per line. Each puzzle blank line begins with the number of letter spaces in the blank, followed by the codes for each letter space.

```
7 4
UNDERFLOW
HARDWARE
ANALYTIC
DATABASE
FUNCTION
STRUCTURE
PROCESSOR
8 69 26 54 96 65 85 49 43
9 29 65 47 26 96 65 46 47 52
9 46 54 59 52 47 69 92 49 23
8 48 54 83 63 79 65 20 60
```

Output

The program must print the solution to the puzzle, one word per line in the same order as the puzzle blanks.

```
FUNCTION
STRUCTURE
UNDERFLOW
ANALYTIC
```

If your program fails, the judges won't be able to give you feedback. Be sure to test your program with all the sample data sets.

Problem.19 The Hoppy Frog

+ 13 points

Summary

A retired military general keeps a very orderly life and chose for her pet a very orderly retired military frog. Every day, the general rearranges the stepping stones within her four ornate ponds, providing a new jumping pattern for the frog.

The stepping stones are arranged in a perfect square grid. One quarter of the stones are in each of the four ponds; that is,

- the stones in the top-left quadrant of the grid are all in one pond,
- the stones in the top-right quadrant of the grid are all in a second pond,
- similarly the other two ponds each hold the stones in the other two quadrants.

The frog is very disciplined and proceeds through his exercises according to specific rules. The general understands this and sets the stones appropriately.

- The frog starts on stone 1, facing clockwise around the four ponds.
- When he jumps, he leaves one pond to land on a stone in the next pond.
- Each jump takes him to the next numbered stone in sequence.
- He jumps in a straight line, landing on a stone in the same row or column as he started.
- When he lands, he makes a right turn to prepare for the next jump.
- Upon reaching the highest number, he turns right and makes one more jump to return to stone 1.

One possible configuration of stones in the four ponds:

```
01 05 | 06 02
09 13 | 14 10
-------------
16 04 | 15 03
08 12 | 07 11
```

After a few weeks, the general decided to make the pattern more complicated. Sometimes she placed 9 stones in each pond (36 total) or 16 stones in each pond (64 total.) And for added complexity, she turned over some stones, hiding their numbers. The veteran amphibian tackled every challenge effortlessly, each day starting on stone 1 and proceeding to land on each stone in the proper order, even when its number wasn't visible, finishing his routine on stone 1.

Are you as perceptive as a regimental frog? Your program must read in the grid of stones, with known and unknown numbers. Then, knowing the jumping rules, your program must determine the location of every number in the grid. There is only one solution.

Input

The first line is the width N in stones of the square grid (4, 6 or 8). The next N lines each hold one row of the grid, with N 2-digit numbers, separated by one space. 00 represents an upside down stone.

```
Example 1:
  4
  01 00 06 00
  00 00 00 10
  00 00 00 03
  08 00 00 00
```

```
Example 2:
  6
  00 00 14 00 00 31
  06 00 00 00 00 00
  00 18 00 00 00 00
  33 00 00 00 00 00
  00 00 00 08 12 36
  00 00 00 04 28 00
```

```
Example 3:
  8
  00 22 58 00 00 00 00 00
  30 50 00 00 00 00 00 47
  02 00 00 00 00 00 35 00
  00 00 06 00 00 00 00 00
  53 00 00 00 24 00 00 00
  00 00 00 00 00 04 44 12
  41 00 00 00 00 20 00 56
  00 61 00 00 64 32 16 00
```

Output

Your program must print the value of every stone, in the same format as the input, 2 digits per stone, separated by one space.

```
Example 1:
  01 05 06 02
  09 13 14 10
  16 04 15 03
  08 12 07 11
```

```
Example 2:
  02 30 14 03 15 31
  06 10 22 07 11 23
  34 18 26 19 27 35
  33 17 21 20 16 32
  01 09 13 08 12 36
  05 29 25 04 28 24
```

```
Example 3:
  42 22 58 18 23 19 43 59
  30 50 26 46 27 31 51 47
  02 62 34 10 63 03 35 11
  54 14 06 38 39 07 15 55
  53 49 25 09 24 08 52 48
  29 13 05 45 28 04 44 12
  41 21 57 37 40 20 36 56
  01 61 33 17 64 32 16 60
```

Problem.20 Bell Ringers

+ 14 points

Summary

Each year to celebrate CodeWars, our official CW Bell Ringers, or "CowBell Ringers" gather for us to ring in the festivities. They're more entertaining, cleaner, and have a better nickname than the CW Pi Throwers we had a few years ago.

Of course, it being CodeWars, they have an unpredictable performance. Each Ringer is assigned a different number of seconds between rings and start their counts at different times. They stand in a line and start a metronome ticking every second. Each Ringer starts counting their seconds when the prior bell ringer rings their bell.

For example, say there are four Ringers, with delays of 5, 3, 7 and 8 seconds.

- When the metronome starts (time=0), the first Ringer starts counting, and when her 5 seconds have elapsed, she rings Bell#1 at time 5, and then rings Bell#1 again every 5 seconds (at 10, 15, 20 ...)
- When the second Ringer hears Bell#1, he starts counting and when his 3 seconds have elapsed, he rings Bell#2. His first ring is at 8 (5+3), then every 3 seconds (11, 14, 17, 20, ...)
- When the third Ringer hears Bell#2, she starts counting and when her 7 seconds have elapsed, she rings Bell#3. Her first ring is at 15 (5+3+7), then every 7 seconds (22, 29, 36, 43, 50 ...)
- The fourth Ringer starts counting at 15 (when Bell#3 rings), then rings Bell#4 every 8 seconds at 23 (5+3+7+8), 31, 39, 47, 55, 63 ...
- With more Ringers, each Ringer is similar, waiting for the prior bell to ring before starting their first count.

The Ringers continue their chorus until all bells ring simultaneously. For the values above, this happens at 575 seconds, a pleasant performance less than 10 minutes.

Input

The input describes several choirs. The first number on a line is N, the number of Ringers (maximum of 8). This is followed by N integers, each between 1 and 50, the number of seconds for each Ringer. The last line of input is a single 0. (In case it helps, the product of all Ringer delays will be less than 100 million.)

```
4 5 3 7 8
8 4 5 16 2 3 7 8 9
3 1 2 3
8 2 6 4 8 3 5 7 1
5 25 24 23 19 17
6 25 24 23 19 17 13
0
```

Output

We need to know how long each show will last. For each line, your program must print the time in seconds when the entire choir rings their bells together and the show ends. There may be assigned delays for which the choir can never ring together. You must identify those and instead print "They go forever!" The maximum time allowed for your program to find the result for each choir is 10 seconds (10 seconds for one line of input), so consider how to be efficient in your search.

```
575
They go forever!
9
308
2215225
37874425
```

Problem.21 Hyperspace Bypasses

+ 15 points

Summary

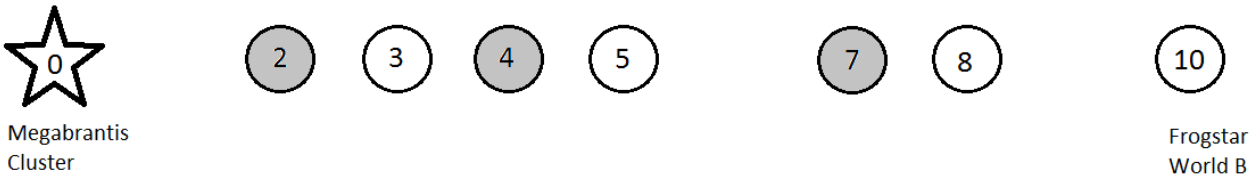
The Galactic Civil Service has decided to expand the travel options from around the galaxy to the Megabrantis cluster. To accomplish this, the service has commissioned the Vogon constructor fleet to build a series of hyperspace bypasses originating from the cluster's core to places of interest across the Milky Way.

Because the bypasses stretch across vast distances, the civil service wants rest stops placed along the route at convenient intervals for travelers. Because the Vogons are in charge of building the bypasses, they insist each rest stop placement result in the destruction of a planet along the route, which always comes as unwelcome news for the inhabitants of these planets. That being the case, rest stops must never displace a planet at the end of the route.

The placement of rest stops along the route should ensure the following distances are as large as possible:

- Distance from the cluster's core to the first rest stop
- Distance from a rest stop to the next rest stop
- Distance from the final rest stop to the planet at the end of the route

Consider the route from the cluster's core to Frogstar World B -- the future site of Milliways, the Restaurant at the End of the Universe. There are planets along the route at distances from the core of 2, 3, 4, 5, 7, 8, and 10. The Vogons are asked to place 3 rest stops on the bypass route. For the greatest minimum distance, the Vogons could build at the planets of distances 2, 4, and 7.



Distance from cluster's core to the first rest stop: 2
Distance from first rest stop to second rest stop: 2
Distance from second rest stop to third rest stop: 3
Distance from the final rest stop and the destination planet: 3

The largest minimum distance between stops is 2. Note that there could be multiple possible placements of rest stops that would result in the same minimum distance between stops. In the example above, rest stops could instead have been placed at planets of diststances 3, 5, and 8.

Given a list of planet locations along a bypass route, output the largest minimum distance between stops (including all rest stops, the cluster core, and the final planet).

Input

Input will begin with an integer T (0 < T <= 20) representing the number of test cases. Each test case will consist of two lines. The first line will contain two integers: P (1 < P <= 100,000) and R (0 < R < P). P represents the number of planets along the bypass route and R represents the number of required rest stops.

The second line will contain P space separated integers D (0 < D <= 1,000,000,000,000,000) representing the distance of each planet from the galactic hub. The distance of each planet will be distinct.

```
6
7 3
2 3 4 5 7 8 10
5 2
100 2 3 1 4
5 2
100 40 20 10 30
2 1
500000000000000 1000000000000000
5 2
200000000000000 400000000000000 600000000000000 800000000000000 1000000000000000
7 3
10 4 7 3 5 2 8
```

Output

For each test case, output a line with the following format:
{Test Case Number}: {Largest Minimum Distance Between Stops}

```
1: 2
2: 2
3: 20
4: 500000000000000
5: 200000000000000
6: 2
```

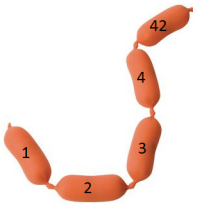
Problem.22 Prime Cuts

+ 16 points

Summary

The migration of Perfectly Normal Beasts across the Anhondo Plain is one of the stranger and more wonderful visions the galaxy has to offer. For the people of Lamuella, this twice yearly event is a time for celebration. The village hunters kill about a dozen of the thousands of beasts thundering past, and there's a feast to celebrate. The meat left over from the feast is salted and stored for the winter. Some of the meat is made into links of sausages.

Arthur Dent, the village Sandwich Maker and caretaker of the meat, likes to put together a winter feast by making sausage pie. Arthur considers a sausage pie **perfect** if the weight of the sausages in the pie is a prime number. (A prime number is a positive integer greater than 1 with no positive integer divisors other than 1 and itself.) A link of sausages is **perfectly prime** if Arthur can make at least one perfect pie and use every sausage in the link, making every perfect pie from contiguous sausages along the link.



Consider a link of sausages with weights of 1, 2, 3, 4, and 42, which is not a **perfectly prime** link of sausages. Arthur could make three perfect pies of the sausages as follows:

Pie 1: 1 and 42
Pie 2: 2
Pie 3: 3 and 4

Although all of the sausages were used, the link is not **perfectly prime** because the sausages of weights 1 and 42 are not contiguous.

Note that **perfectly prime** links of sausages can have multiple possible combinations of sausages used to make the perfect pies. Consider a link with sausages of weights 37, 4, 3, 2, and 1. The following combinations of sausages can be used:

Combination 1: 37; 4 and 3; 2 and 1
Combination 2: 37 and 4; 3; 2 and 1
Combination 3: 37, 4, 3, 2, and 1

Given the weights of sausages in a link, output whether the link is **perfectly prime**. If a link is **perfectly prime**, show one possible combination of sausages used for each pie.

Input

Input will begin with an integer T (0 < T <= 21) representing the number of test cases. The following T lines will begin with an integer L (0 < L <= 42) representing the number of Perfectly Normal Beast sausages in a link. The rest of the line will contain L space-separated integers representing the weight of each sausage. The weight of each sausage will not exceed 500,000.

```
12
5 1 2 3 4 42
7 1 2 3 4 5 6 7
5 37 4 3 2 1
8 1 2 3 4 5 6 7 8
8 8 1 2 3 4 5 6 7
1 2
2 3 4
41 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 16142
42 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 16142
41 492080 3 5 2 2 3 2 3 2 5 2 2 5 3 2 2 5 2 2 3 2 2 3 3 2 3 5 2 2 3 2 3 2 3 2 3 2 3 2 2 492175
42 3 2 3 2 2 5 2 3 2 5 5 2 2 3 2 2 2 3 2 2 3 2 2 5 2 5 2 3 2 2 5 2 5 2 2 5 3 2 2 370274 2 3
42 360637 3 2 2 3 2 3 2 3 2 5 2 5 2 2 5 2 2 3 2 3 2 5 2 3 2 5 2 5 3 5 5 3 3 3 2 5 2 3 2 3 370267
```

Output

For each test case, output a line with the following format:

- Test case number followed by a colon
- Whether the link is perfectly prime
- For perfectly prime links, a space-separated list of the weight of each sausage in the link with a vertical bar character indicating the separation point for each perfect pie. If a perfect pie can be made from the entire link, no vertical bar is needed.

Note: Some links will have multiple combinations of valid separation points between pies. Your separation points don't need to match the sample output, but they must be valid.

```
1: No
2: Yes 1 2 | 3 4 | 5 | 6 7
3: Yes 37 4 3 2 1
4: No
5: Yes 8 1 2 3 4 5 6 | 7
6: Yes 2
7: Yes 3 4
8: No
9: Yes 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 16142
10: No
11: No
12: Yes 360637 | 3 2 2 3 2 3 2 3 2 5 2 5 2 2 5 2 2 3 2 3 2 5 2 3 2 5 2 5 3 5 5 3 3 3 2 5 2 3 2 3 370267
```

Problem.23 Heart of Gold

+ 19 points

Summary

Tired of a lot of tedious mucking about in hyperspace to travel vast interstellar distances, the Imperial Galactic Government is working on a prototype ship -- the Heart of Gold -- that uses the revolutionary Infinite Improbability Drive.

The prototype ship is still in its earliest stages (travel is only possible within a planet's atmosphere) and it has a key limitation that researchers are still working through:

- The ship can travel infinitely fast when traveling over land but takes one unit of time to cross from an edge to any other edge over a 1-by-1 square grid of water.

The government is conducting its research in the Damogran system, a series of planets consisting of nothing but middling to large desert islands separated by very pretty but annoyingly wide stretches of ocean.

The islands and oceans of the planets in the Damogran system have an unusual layout -- a grid of 1-by-1 squares of water or land whose edges run parallel to the four cardinal directions (north, south, east, and west). Each square of land that makes up an island shares at least one full edge with another square of the island. Squares of land from separate islands have at least one square of water between one another on all edges and diagonally.

An island will contain no more than 500 squares of land. To aid travelers, a network of navigation buoys run north to south and east to west through the center of each square of ocean on a planet. Vessels travelling on, above, or below the ocean waters must follow the buoys on a north/south or east/west heading.

Given a map of a planet and a list of starting and ending coordinates, output the starting and ending island along with the minimum travel time between the two islands on the Heart of Gold.

Input

Input will begin with a pair of integers R and C (3 <= R, C <= 200) representing the number of rows and columns of a planet's map. The first character of the first line is at coordinates 0, 0 and is the northwest corner of the map.

The following R lines will contain a string C characters long consisting of either a **.** character (representing ocean) or an uppercase or lowercase letter representing land. Letters of the same character (case sensitive) make up an island.

The next line will contain an integer Q (0 < Q <= 20) representing the number of questions.

The following Q lines will contain four space-separated integers X1, Y1, X2, Y2 (0 <= X1, X2 < R; 0 <= Y1, Y2 < C) representing the starting and ending coordinates for a trip in the Heart of Gold. 'X' coordinates indicate the row, and 'Y' coordinates indicate the column. The starting and ending coordinates are guaranteed to be on an island.

```
10 35
.......oo..................uuuuuuuu
..A...oooo.rr......B.y.....uu..u.uu
......o.........l..................u
.j.Q..oo........l..m.s...iiiii..uuu
.j.Q...............m.s...i...i...uu
.j.Q..W............m.s...i.g.i....u
.j.Q..W.........d..m.s...i...i....u
......W.........d........iiiii...uu
..a...W............P.z.............
.........................cccccc...
8
1 2 8 2
3 7 1 9
7 6 7 16
5 6 7 16
5 27 3 25
1 19 8 21
1 19 8 19
3 29 7 25
```

Output

For each question, output a single line that shows the starting island, ending island, and minimum travel time. Use the following format: "{start} to {end} takes {time}"

```
A to a takes 4
o to o takes 0
W to d takes 8
W to d takes 8
g to i takes 1
B to z takes 3
B to P takes 2
i to i takes 0
```