1. Fastest Animals

Standard Bank

Introduction

Which animal in the zoo is the fastest? Every schoolchild will tell you that it is the cheetah. But is it? In this modern world of fake news and alternative facts, the one way to be sure is to test it yourself.

Task

The zookeepers have timed the zoo's animals running over various distances. For each animal, they know the distance the animal covered and how long it took. They need your help to compute the speeds of the animals.

A program which deals with the input and output of this problem has already been written, but it needs to be modified to implement a function which does the computation. You can download the source code for this program, in your preferred language, from the *Resources* tab in Abacus.

Note: *you may instead write a program from scratch, but if you have not taken part in the Technology Impact Challenge or a similar contest before, you are advised to use the example code as it will correctly handle the details of input and output. Do not add any input or output statements to the program as doing so will prevent the automated marker from marking it correctly.*

Example

The cheetah covered 120 m in 4 sec while the hare covered 34 m in 2 sec. Thus, the cheetah's speed is $30\,\mathrm{m\,s^{-1}}$ and the hare's is $17\,\mathrm{m\,s^{-1}}$.

Input

The input describes multiple animals. The first line contains T, the number of animals. The remaining T lines each describe one animal. An animal is described by two numbers, m and s, separated by a space, indicating that m metres were covered in s seconds. Conveniently, m is always an exact multiple of s.

Sample input

```
2
120 4
34 2
```

Output

For each test case, output a line of the form

```
Case #X: r
```

where X is the animal number, starting from 1, and r (an integer) is the speed of the animal in metres per second.

Sample output

```
Case #1: 30
Case #2: 17
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 10$
- $1 \leq m \leq 1\,000$
- $1 \leq s \leq 1\,000$
- m is an exact multiple of s

Time limit

2 seconds

2. Bodmas Bonobos

Standard Bank

Introduction

The Bodmas bonobo is a rare and brilliant breed of chimpanzee. These apes learn from a young age how to perform calculations involving addition, subtraction and multiplication. They are called *Bodmas* bonobos because they have adopted the same convention as humans for the order of operations, namely BODMAS. That is, when evaluating an expression, they perform the multiplications before performing the additions and subtractions.

The zoo has recently been fortunate enough to acquire a troop of Bodmas bonobos. Barty, the youngest of the troop, has been having some difficulty mastering the BODMAS rules. Although he is able to perform addition, subtraction and multiplication as well as the other bonobos, he simply evaluates expressions from left to right, ignoring the usual order of operations.

Barty's elder siblings have been using this to their advantage. Whenever food is to be divided between the children, they represent their shares as mathematical expressions. However, they devise these expressions to fool Barty into thinking that each bonobo will get a different amount than it will actually get.

Task

Given two numbers A and B, construct a single expression which evaluates to A when applying the usual BODMAS order of operations, but evaluates to B when calculating from left to right as Barty would.

Example

Suppose $A = 9$ and $B = 6$. Then we may construct the expression $0 - 3 + 6 * 2$, which simplifies to $-3 + 6 * 2 = -3+12 = 9$ when using the BODMAS order of operations, but simplifies to $-3+6*2 = 3*2 = 6$ when performing the operations from left to right. Another possible expression is $1 - 2 * 2 + 4 * 3$, which simplifies to $1 - 4 + 4 * 3 = -3 + 4 * 3 = -3 + 12 = 9$ when applying the BODMAS rules, but simplifies to $-1*2+4*3 = -2+4*3 = 2*3 = 6$ when operating from left to right as Barty would.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case consists of a line containing the space-separated integers A and B.

Sample input

```
5
9 6
5 5
4 2
1 0
5 30
```

Output

For each test case, output a line of the form

```
Case #X: E
```

where X is the test case number, starting from 1, and E is the expression which is to be evaluated. The expression E must have the form

```
x1 o1 x2 o2  ... oN-1 xN
```

where the x_i's are integers satisfying $0 \leq x_i \leq 10\,000$, each of the o_i's is one of "`+`", "`-`" or "`*`", and the number of integers N is bounded by $1 \leq N \leq 10$. Note that there are many correct answers of this form for any given values of A and B, and any one of them will be accepted.

Sample output

```
Case #1: 9 - 6 + 0 * 0 + 6
Case #2: 5 - 5 + 0 * 0 + 5
Case #3: 4 - 2 + 0 * 0 + 2
Case #4: 1 - 0 + 0 * 0 + 0
Case #5: 5 - 30 + 0 * 0 + 30
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $0 \leq A \leq 100$
- $0 \leq B \leq 100$

Time limit

2 seconds

3. Elephant Encryption

Standard Bank

Introduction

The elephants are planning their escape from the zoo! These clever creatures have decided to document their plans by scribbling messages in the sand using their trunks. In order to prevent their plans from being foiled by the zookeepers, they have developed a top secret encryption algorithm that they use to scramble their messages.

Before writing down a word, an elephant first picks a postive integer K, which it will never forget. It then consecutively applies two distinct procedures to the word, namely *shift-K* and *rotate-K*.

The shift-K procedure involves repeatedly moving the last letter of the word to the beginning of the word, K times in total.

For the rotate-K procedure, the elephant repeatedly replaces each letter of the word with the letter that occurs immediately after it in the alphabet, and does so K times for each letter. For this procedure, the elephants consider the letter that occurs immediately after 'Z' to be 'A'.

Task

Given a word W consisting of L letters and an integer K, determine the encrypted word obtained by consecutively applying the shift-K and the rotate-K procedures to W.

Example

Suppose W is the word "ATTACK" and $K = 4$. Applying shift-K to W yields "TACKAT", and applying rotate-K to this word yields "XEGOEX".

Suppose instead W is the word "RETREAT" and $K = 10$. The word obtained by applying shift-K to W is "EATRETR", which becomes "OKDBODB" after applying rotate-K.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case consists of a line containing the word W and the integer K, separated by a space.

Sample input

```
4
ATTACK 4
RETREAT 10
TRUMPET 99999
STAMPEDE 256
```

Output

For each test case, output a line of the form

```
Case #X: E
```

where X is the test case number, starting from 1, and E is the encrypted word.

Sample output

```
Case #1: XEGOEX
Case #2: OKDBODB
Case #3: PSHWWUX
Case #4: OPWILAZA
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- W consists of uppercase letters from 'A' to 'Z'
- $1 \leq L \leq 10\,000$
- $1 \leq K \leq 10^9$

Time limit

2 seconds

4. Travelling Salesman

Standard Bank

Introduction

A travelling animal-feed salesman is planning to make a trip to South Africa, and he wants to visit every zoo in the country exactly once. While it's not something the zoos advertise much, it is possible to fly from one zoo to another in a bag carried by a flock of African swallows. In some countries the salesman's goal of visiting every zoo once without re-visiting any zoo is complicated by the lack of direct flights, but he has heard that in South Africa, every pair of zoos is linked by a direct flight. "At last," he thinks, "a civilised country that won't force me to spend hours working on my schedule."

Unfortunately, at the last minute he discovers that while every pair of zoos is linked by a direct flight, these flights are all one-way: for two zoos A and B, it will be possible to fly directly from A to B or from B to A, but never both.

Task

Help the salesman plan the order in which he should visit the zoos. Produce a list in which every zoo appears exactly once, and in which there is a direct flight from each zoo (other than the last) to the next zoo on the link.

Example

Consider four zoos, with direct flights $1 \to 3$, $2 \to 1$, $2 \to 4$, $3 \to 2$, $4 \to 1$ and $4 \to 3$. One way the salesman can achieve his goal is to travel $2 \to 4 \to 1 \to 3$.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case starts with a line containing N, the number of zoos, which are numbered from 1 to N. This is followed by N lines each containing N characters. The jth guarantee of the ith line is `Y` if there is a direct flight from zoo i to zoo j, and `N` otherwise.

It is guaranteed that there is no flight from a zoo to itself, and for two distinct zoos, there is a direct flight in exactly one direction between them.

Sample input

```
2
4
NNYN
YNNY
NYNN
YNYN
5
NNYYN
YNNNY
NYNYN
NYNNY
YNYNN
```

Output

For each test case, output a line of the form

```
Case #X: z1 z2 ... zN
```

where X is the test case number, starting from 1, and $z_1, z_2, \ldots, z_N$ are the numbers of the zoos in the order that the travelling salesman should visit them.

You may assume that an answer always exists. If there are multiple possible answers, you may output any of them.

Sample output

```
Case #1: 2 4 1 3
Case #2: 2 1 3 4 5
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 15$
- $2 \leq N \leq 500$

Time limit

2 seconds

5a. Cave Expansions

Standard Bank

Introduction

A large underground cave system has recently been discovered on the property of the zoo. The caves would make a great new home for the zoo bats, who are currently living in cages above the ground. Before relocation of the bats can begin, the zookeepers need to determine how many caves there are, and how large each cave is.

The zookeepers have employed the use of a fancy radar device to survey the top-down layout of the cave system. This layout is displayed on the device as a two-dimensional grid, with each cell in the grid indicating whether the position corresponding to that cell is *open* (part of a cave) or *blocked* (earth all the way down).

Since bats mostly stick to ceilings, the heights of the caves are not important, and this two-dimensional map is sufficient for determining the sizes of the caves.

Task

Given a two-dimensional grid detailing the top-down layout of the cave system, determine how many caves there are, and how many open cells each cave consists of. Two open cells are considered part of the same cave if it is possible to travel between the cells with a sequence of horizontal and vertical moves (i.e. no diagonal moves), without passing through any blocked cells.

Example

Suppose we are given the following two-dimensional grid.

```
#..#
.##.
..##
#.#.
```

There are 4 caves in this system, with sizes 1, 1, 2 and 4.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case starts with a line containing the space-separated integers R and C. The following R lines then describe the layout of the cave system. Each of these lines consists of C characters, each of which is either "`.`", indicating an open cell, or "`#`", indicating a blocked cell.

Sample input

```
4
4 4
#..#
.##.
..##
#.#.
7 7
####.##
..#.##.
#...#..
##.##.#
..#...#
##..#.#
###...#
4 14
#.#########.##
#..###..####.#
###.###...##.#
##..#########.
2 5
#####
#####
```

Output

For each test case, output a line of the form

```
Case #X: S1 S2 ... SN
```

where X is the test case number, starting from 1, and the S_i's are the sizes of the caves, sorted in ascending order.

Sample output

```
Case #1: 1 1 2 4
Case #2: 1 2 7 13
Case #3: 1 1 2 3 3 5
Case #4:
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq R \leq 400$
- $1 \leq C \leq 400$

Time limit

2 seconds

5b. Cave Expansions

Standard Bank

Introduction

Soon after they started planning the relocation of the zoo bats, the zookeepers received another instruction from their manager. The investors would like to have one enormous cave which is to be used as the main attraction for tours of the bat caves.

The problem is that none of the caves are quite big enough to satisfy the investors. The manager has provided the zookeepers with some explosives in order to "renovate" the caves, but it will only be enough to blow up a single blocked cell on the two-dimensional grid. Thus, they have to use the explosives wisely.

Task

Given a two-dimensional grid of open and blocked cells, determine the size of the largest cave that can be created by blowing up a single blocked cell, i.e. turning it into an open cell. The cell which is blown up should be included in the size of the new cave.

Example

Suppose we are given the following two-dimensional grid.

```
#.##..
.#..#.
#.#.#.
..#..#
```

The largest possible cave has size 11, and can be created by blowing up the blocked cell in row 2, column 2.

Input

The input format remains the same.

Sample input

```
3
4 6
#.##..
.#..#.
#.#.#.
..#..#
2 5
.##..
.###.
4 9
..#..####
##..#..#.
..##.#.#.
.##..##..
```

Output

For each test case, output a line of the form

```
Case #X: S
```

where X is the test case number, starting from 1, and S is the size of the largest possible cave.

Sample output

```
Case #1: 11
Case #2: 4
Case #3: 11
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq R \leq 400$
- $1 \leq C \leq 400$
- There is at least one blocked cell in each grid.

Time limit

4 seconds

6. Mixing Drinks

Standard Bank

Introduction

After work the zookeepers like to go to the bar for a relaxing drink. The barman is highly qualified and knows how to mix just about any cocktail you could name. Unfortunately, his talent is entirely wasted on the zookeepers. They are not fussy at all, and care only about the ratio of water to alcohol in their drinks.

Task

The bar stocks a number of ingredients. Each ingredient has a known ratio of water to alcohol. Of course, they also contain other things, but these are in much smaller amounts and should be ignored for this problem.

Each zookeeper states his or her order as a desired ratio of water to alcohol. Help the barman determine whether each order can be fulfilled by mixing together ingredients, and if so, find one way to do so.

Example

Suppose the bar stocks wine (8 : 1 ratio), brandy (4 : 1) and vodka (1 : 2). A ratio $a : b$ means that the drink contains a parts water to b parts alcohol.

Three zookeepers are at the bar, and request drinks with strengths 19 : 6, 5 : 4 and 31 : 3. The first drink can be made by mixing 3 parts wine, 1 part brandy and 1 part vodka: Table 1 shows how this might work if a "part" is 45 ml. The second drink can be made by mixing 2 parts wine and 3 parts vodka. There are also other ways these drinks can be mixed. However, the last order cannot be made by mixing the ingredients in any combination.

Table 1: Mixing a 19 : 6 drink in the first example

Ingredient	Parts	Quantity (ml)	Water (ml)	Alcohol (ml)
Wine	3	135	120	15
Brandy	1	45	36	9
Vodka	1	45	15	30
Total		225	171	54

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case starts with a line containing two integers N and M, separated by a space. There are N ingredients and M zookeepers.

The next N lines describe the ingredients, one per line. Each line contains two integers a_i and b_i, separated by a space, indicating an $a_i : b_i$ ratio of water to alcohol.

The following M lines describe the zookeepers' orders, one per line. Each line contains two integers c_j and d_j, separated by a space, indicating an order with a $c_j : d_j$ ratio of water to alcohol.

Sample input

```
2
3 3
8 1
4 1
1 2
19 6
5 4
31 3
2 3
4 3
8 6
1 1
2 1
12 9
```

Output

For each test case, output a line of the form

```
Case #X:
```

where X is the test case number, starting from 1. This must be followed by M lines, one for each drinks order, in the same order they appeared in the input file. If it is impossible to make up the order from the ingredients, output a line containing only the word `NO`. Otherwise, output a line of the form

```
YES p1 p2 p3 ... pN
```

This indicates that the drink can be obtained by mixing p_i parts of the ith ingredient from the input file. Some (but not all) of the p_i may be zero. The sum of all the p_i must not exceed 10^9. If there are multiple ways to mix a drink, you may output any of them.

Sample output

```
Case #1:
YES 3 1 1
YES 2 0 3
NO
Case #2:
NO
NO
YES 1 0
```

6. Mixing Drinks

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq N \leq 50$
- $1 \leq M \leq 50$
- $1 \leq a_i, b_i, c_i, d_i \leq 50$

Time limit

2 seconds

7. Making a Hash of it

Standard Bank

Introduction

Sometimes mischievous visitors to the zoo make humorous but unwanted changes to the signs, such as changing the "Do not feed the monkeys" sign to "Do not feed the monkeys to the lions." These changes are quickly detected due to the use of a *hash function* to verify signs. A *hash function* is a function that processes a string to produce a number in a fixed range. The particular hash function used by the zoo is described later.

When a sign is installed, the zoo computes the hash of the text on the sign and records it. Every morning, they again compute the hash of each sign, and if the newly computed value doesn't match the recorded value, then they know that the sign has been monkeyed with.

Of course, if a visitor can come up with a string that has the same hash value as the original, then the change will never be detected. After hearing about recently-discovered flaws in a well-known hash function (SHA-1), the zoo has hired you as a security consultant to test their hash function.

Task

You will be given the hash values of the zoo's signs. For each sign, you must find a string that has this hash value.

The hash function used by the zoo is described below. Implementations in the allowed programming languages can be found on the Resources tab. The function operates on lower-case English letters:

1. Set h to 981469373221.
2. For each character in the string:
   (a) Multiply h by 435, divide by 2^{40}, and replace h with the remainder.
   (b) Set h to h XOR c, where c is the ASCII value of the character (97 for 'a' through to 122 for 'z').

Example

Here is the calculation of the hash value of "zoo".

Operation	h
Initialise	981469373221
$h \leftarrow h \times 435 \mod 2^{40}$	328665774047
$h \leftarrow h$ xor 'z'	328665773989
$h \leftarrow h \times 435 \mod 2^{40}$	33100074335
$h \leftarrow h$ xor 'o'	33100074288
$h \leftarrow h \times 435 \mod 2^{40}$	104881154192
$h \leftarrow h$ xor 'o'	104881154303

Input

The first line of input contains T, the number of signs. The remaining T lines each contain an integer h, the hash value of a sign.

Sample input

```
3
104881154303
95308969361
484660377506
```

Output

For each test case, output a line of the form

```
Case #X: M
```

where X is the sign number, starting from 1, and M is a string of lower-case English letters whose hash value is given in the corresponding line of input. M must contain between 1 and 1000 characters, inclusive. Note that there are multiple valid outputs; you may output any of them.

Sample output

```
Case #1: zoo
Case #2: infinitemonkeys
Case #3: qemrrgmwergmewrgfij
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 10$
- $0 \leq h < 2^{40}$

Time limit

20 seconds