Submissions

Ticket Trouble

5pt Not attempted **613/819 users** correct (75%)

10pt Not attempted **565/616 users** correct (92%)

Understudies

5pt Not attempted **505/616 users** correct (82%)

15pt Not attempted **440/508 users** correct (87%)

Word Search

10pt Not attempted **186/337 users** correct (55%)

15pt Not attempted **20/77 users** correct (26%)

Where Ya Gonna Call?

15pt Not attempted **23/91 users** correct (25%)

25pt Not attempted **3/22 users** correct (14%)

Top Scores

Taube	100
ponik	75
aquannie	75
YuryBandarchuk	75
Penguinsheaven	75
Marjan0003	75
MiriTheRing	75
Celicath	60
n.bezrodnaya	60
FireJade	60

Problem A. Ticket Trouble

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 5 points — Solve A-small

Large input 10 points — Solve A-large

Problem

A group of **F** friends is attending a conference held at an amphitheater, and they have bought tickets to see a concert there afterwards. The amphitheater is a grid of seats with **S** rows and **S** columns. For each seat, the amphitheater has sold a single ticket (although some of the tickets might not have been sold to this group of friends). Each ticket is normally labeled with a pair of integers giving the row and column numbers of one seat, in that order. For example, a ticket might normally say (2, 1), meaning row 2, column 1, or (2, 2), meaning row 2, column 2.

When the tickets were printed, there was a malfunction, and the two numbers in each pair always came out in sorted (that is, nondecreasing) order! So, for example, a ticket labeled (1, 2) might actually be for the seat in row 1, column 2, or it might actually be for the seat in row 2, column 1. If two friends have tickets labeled (1, 2), then one must actually be for row 1, column 2, and the other must actually be for row 2, column 1.

The friends will consult the box office on the day of the concert to find out what their actual seat numbers are, but for now, it is unclear! Given the printed pairs on the tickets, what is the largest possible number of the friends that could actually be seated all in the same-numbered row of seats? (The friends do *not* necessarily need to be seated in consecutive seats in that row.)

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each begins with one line with two integers **F** and **S**, representing the number of friends and the dimension of the grid of seats. Then, **F** more lines follow. The i-th of those lines has two integers $\mathbf{A_i}$ and $\mathbf{B_i}$, representing the two numbers printed on the i-th friend's ticket.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is largest possible number of the friends that could actually be seated all in the same-numbered row of seats.

Limits

$\mathbf{F} \le \mathbf{S}^2$.
$1 \le \mathbf{A_i} \le \mathbf{B_i} \le \mathbf{S}$, for all i.
No pair appears more than twice in a test case.
No pair containing the same number twice appears more than once in a test case.

Small dataset

$1 \le \mathbf{T} \le 50$.
$2 \le \mathbf{F} \le 3$.
$2 \le \mathbf{S} \le 3$.

Large dataset

$1 \le \mathbf{T} \le 100$.
$2 \le \mathbf{F} \le 100$.
$2 \le \mathbf{S} \le 100$.

Sample

Input	Output
3 2 3 1 2 1 2 3 3 1 2 2 3 2 2 3 3	Case #1: 1 Case #2: 3 Case #3: 2

```
1 1
2 2
1 2
```

In sample case #1, one ticket must actually be for row 1, column 2, and the other must actually be for row 2, column 1, even though we do not know which is which. So we know that the friends are not seated in the same row, and the largest number of friends in any row is 1. Also note that the seats have a third row and column, but none of the tickets use the third row or column.

In sample case #2, one of the tickets is definitely for seat 2 in row 2, and it is possible that two of the other tickets could be for seats 1 and 3 in row 2. So there may be as many as 3 friends in the same row.

In sample case #3, either there are two friends in row 1 and one in row 2, or there are two friends in row 2 and one in row 1. In either case, the answer is 2.

Problem B. **Understudies**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 5 points — Solve B-small

Large input 15 points — Solve B-large

Problem

You are a casting director for an upcoming musical. The musical has **N** roles, and for each role, you want to cast two performers: one primary performer and one understudy. A primary performer or understudy trains for only one particular role, and the job of the understudy is to play the role if the primary performer becomes unavailable. At least one of the two performers for each role must be available for the show to succeed.

You have selected 2**N** performers to be in the musical. They are all quite talented, and any of them can be cast as a primary performer or understudy for any of the roles. However, you are worried that some of them may be tempted to run away to join the cast of *Hamiltonian!*, the smash hit musical about quantum mechanics, before your show opens. Luckily, you are an excellent judge of character. You know that the i-th performer has a probability $\mathbf{P_i}$ of becoming unavailable. (These probabilities are all independent of each other, and a given performer has their probability regardless of their assigned role or whether they are a primary performer or understudy.)

You wish to assign one primary performer and one understudy for each role in a way that maximizes the probability that the show will succeed. That is, you want to minimize the probability that there will be at least one role for which the primary performer and understudy both become unavailable.

If you make optimal casting choices, what is the probability that your show will succeed?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of two lines. The first line contains a single integer **N**: the number of roles. The second line contains 2**N** rational numbers $\mathbf{P_i}$; the i-th of these gives the probability that the i-th performer will become unavailable for your show. All of these probabilities are specified to exactly four decimal places of precision.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the probability that your show will succeed. y will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

$1 \le \mathbf{T} \le 100$.
$0.0000 \le \mathbf{P_i} \le 1.0000$, for all i.

Small dataset

$1 \le \mathbf{N} \le 4$.

Large dataset

$1 \le \mathbf{N} \le 40$.

Sample

Input

```
3
2
0.2500 0.5000 0.5000 0.2500
3
0.0000 0.0000 0.0000 0.0009 0.0013 0.1776
1
1.0000 0.1234
```

Output

```
Case #1: 0.765625
Case #2: 1.000000
Case #3: 0.876600
```

In sample case #1, one optimal casting choice is to make the two 0.5000 performers leads for the two roles, and the two 0.2500 performers understudies. For a given role, the probability that both performers will become unavailable is 0.5 × 0.25 = 0.125. So the probability that a role will be filled by at least one of its actors is 1 - 0.125 = 0.875. The probability that both roles will be filled (and thus that the show will succeed) is 0.875 × 0.875 = 0.765625.

If we instead cast the two 0.5000 performers for one role and the two 0.2500 performers for the other role, the probability of success would be (1 - 0.50 × 0.50) × (1 - 0.25 × 0.25) = 0.703125, which is lower.

In sample case #2, the show will succeed for sure as long as you cast exactly one of the 0.0000 performers (who will never become unavailable) in each role.

In sample case #3, the 1.0000 performer will always become unavailable, so the probability of success is equal to 1 minus the probability that the other performer will become unavailable.

Problem C. Word Search

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 10 points — Solve C-small

Large input 15 points — Solve C-large

Problem

In honor of Google I/O 2017, we would like to make an I/O-themed word search grid. This will be a rectangular grid in which every cell contains one of the three characters `I`, `/`, or `O`. The people solving our word search will look for all instances of the string `I/O` that appear contiguously forwards or backwards in a row, column, or diagonal. For example, the following grid contains eight instances of `I/O`, representing all eight possible directions in which the string can appear:

```
OOOOO
O///O
O/I/O
O///O
OOOOO
```

To control the difficulty level of our word search, we would like the string to appear *exactly* **N** times in the grid. Moreover, we do not want the grid to be too large; it cannot have more than **D** rows or more than **D** columns.

Can you help us design a grid that meets these specifications?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with two integers **D** and **N**, as described above.

Output

For each test case, first output one line containing `Case #x:`. Then output R lines of exactly C characters each, representing the rectangular grid. Each of those characters must be either `I`, `/`, or `O`. You may choose any values of R and C as long as both are at least 1 and neither exceeds **D**. Your grid must contain *exactly* **N** instances of the string `I/O`, per the rules described in the statement.

If there are multiple valid answers, you may output any of them.

Limits

$0 \le \mathbf{N} \le 287$.
It is guaranteed that at least one valid grid exists for each test case.

Small dataset

$1 \le \mathbf{T} \le 25$.
$\mathbf{D} = 50$.

Large dataset

$1 \le \mathbf{T} \le 100$.
$\mathbf{D} = 15$.

Sample

Input	Output
4 50 1 50 0 50 3 50 8	Case #1: O / I Case #2: IO Case #3: IIIOOO /I/O/O IIIOOO Case #4: OOOOO O///O O/I/O O///O

```
00000
```

The sample output displays one set of answers to the sample cases. Other answers may be possible. Note that these cases would only appear in the Small dataset.

Problem D. Where Ya Gonna Call?

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 15 points — Solve D-small

Large input 25 points — Solve D-large

Problem

Gooli is a huge company that owns **B** buildings in a hilly area. The buildings are numbered from 1 to **B**.

Last year, they built a set of slides between buildings that are now the favorite form of transportation between buildings. Slides have been upgraded with suction technology to make them two-way, so a slide between two buildings can be used to travel between those buildings in either direction. Some slides were built with turns, so their lengths do not necessarily follow common sense; for instance, they do not necessarily comply with the triangle inequality. Also, there is at most one slide between any pair of buildings.

Gooli is going to choose a location to install a special super secure phone for the CEO to talk to other important people. They want to minimize the distance by slide from any building to the meeting location, so as to minimize the time that it would take the CEO to reach it from any building. Gooli does not have any more carbon kilotubes to build more slides, and the CEO refuses any other type of transportation, so Gooli's communication security team needs to find the best location that is reachable using only already existing slides. The location could be in a building or a point somewhere within a slide.

When traveling using the slides, the CEO may use a slide, arrive at a building, then use a slide that starts there, arrive at another building, and so on, until she arrives at the desired location. Slides used from end to end contribute their full length to the total distance. If the CEO enters a slide and stops inside it because she found the phone, on the other hand, only the used part of the slide contributes to the total distance. When measuring distance, only the slide distance is important. Distance traveled within buildings to connect to a new slide or reach the phone is considered to be zero.

Given the buildings and slides in existence, can you find any optimal location for the super secure phone and return the distance from a farthest building to it? Note that the distance is the same for any optimal location.

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each test case starts with one line with a single integer **B**, the number of buildings on Gooli's campus. Then, **B** - 1 lines follow. For i = 2, 3, ..., **B**, the (i-1)-th of these lines contains (i-1) integers $\mathbf{D}_{i1}$, $\mathbf{D}_{i2}$, ..., $\mathbf{D}_{i(i-1)}$. $\mathbf{D}_{ij}$ is -1 if there is no slide between the i-th building and the j-th building, or the length of that slide otherwise. All buildings are reachable from any other building using only slides, possibly passing through intermediate buildings.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the distance from an optimal location for the phone to a building farthest from it. y will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

$1 \le \mathbf{T} \le 100$.
$2 \le \mathbf{B} \le 50$.
All buildings are reachable from any other building using only slides, possibly passing through intermediate buildings.
$\mathbf{D}_{ij} \ne 0$, for all i, j.

Small dataset

$-1 \le \mathbf{D}_{ij} \le 2$, for all i, j.

Large dataset

$-1 \le \mathbf{D}_{ij} \le 10^9$, for all i, j.

Sample

```
Input       Output

4           Case #1: 1.500000
3           Case #2: 1.000000
-1          Case #3: 2.500000
1 2         Case #4: 8.500000
3
1
1 1
3
4
2 3
4
9
10 7
7 -1 5
```

Note that the last two cases would not appear in the Small dataset.

In Case #1, all buildings are in a line. The only optimal location is of course the middle point of the line, as any other location would make one of the buildings at the end of the line be farther away.

Case #2 depicts an equilateral triangle. Any of the three buildings would be an optimal location for the phone.

Case #3 is also a triangle, but with sides of different lengths. If we pick any building, the farthest building would be at distance at least 3 from it. On the other hand, if we choose a location inside the slide of size 3, at distance 0.5 from building 3, the distance to a farthest building is improved to 2.5.

In Case #4, the optimal location is inside the slide of length 10 between buildings 1 and 3, at distance 1.5 from building 3 and distance 8.5 from building 1.