Problem A. Country Leader

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 4 points — Solve A-small

Large input 7 points — Solve A-large

Problem

The Constitution of a certain country states that the leader is the person with the name containing the greatest number of different alphabet letters. (The country uses the uppercase English alphabet from A through Z.) For example, the name `GOOGLE` has four different alphabet letters: E, G, L, and O. The name `APAC CODE JAM` has eight different letters. If the country only consists of these 2 persons, `APAC CODE JAM` would be the leader.

If there is a tie, the person whose name comes earliest in alphabetical order is the leader.

Given a list of names of the citizens of the country, can you determine who the leader is?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line with an interger **N**, the number of people in the country. Then **N** lines follow. The i-th line represents the name of the i-th person. Each name contains at most 20 characters and contains at least one alphabet letter.

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 name of the leader.

Limits

1 ≤ **T** ≤ 100.
1 ≤ **N** ≤ 100.

Small dataset

Each name consists of at most 20 characters and only consists of the uppercase English letters A through Z.

Large dataset

Each name consists of at most 20 characters and only consists of the uppercase English letters A through Z and ' '(space).
All names start and end with alphabet letters.

Sample

Input

```
2
3
ADAM
BOB
JOHNSON
2
A AB C
DEF
```

Output

```
Case #1: JOHNSON
Case #2: A AB C
```

In sample case #1, `JOHNSON` contains 5 different alphabet letters('H', 'J', 'N', 'O', 'S'), so he is the leader.

Sample case #2 would only appear in Large data set. The name `DEF` contains 3 different alphabet letters, the name `A AB C` also contains 3 different alphabet letters. `A AB C` comes alphabetically earlier so he is the leader.

Problem B. **Vote**

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 8 points	Solve B-large

Problem

A and B are the only two candidates competing in a certain election. We know from polls that exactly **N** voters support A, and exactly **M** voters support B. We also know that **N** is greater than **M**, so A will win.

Voters will show up at the polling place one at a time, in an order chosen uniformly at random from all possible (**N** + **M**)! orders. After each voter casts their vote, the polling place worker will update the results and note which candidate (if any) is winning so far. (If the votes are tied, neither candidate is considered to be winning.)

What is the probability that A stays in the lead the entire time -- that is, that A will always be winning after every vote?

Input

The input starts with one line containing one integer **T**, which is the number of test cases. Each test case consists of one line with two integers **N** and **M**: the numbers of voters supporting A and B, respectively.

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 A will always be winning after every vote.

y will be considered correct if y 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 ≤ **T** ≤ 100.

Small dataset

0 ≤ **M** < **N** ≤ 10.

Large dataset

0 ≤ **M** < **N** ≤ 2000.

Sample

Input	Output
2 2 1 1 0	Case #1: 0.33333333 Case #2: 1.00000000

In sample case #1, there are 3 voters. Two of them support A -- we will call them A1 and A2 -- and one of them supports B. They can come to vote in six possible orders: A1 A2 B, A2 A1 B, A1 B A2, A2 B A1, B A1 A2, B A2 A1. Only the first two of those orders guarantee that Candidate A is winning after every vote. (For example, if the order is A1 B A2, then Candidate A is winning after the first vote but tied after the second vote.) So the answer is 2/6 = 0.333333...

In sample case #2, there is only 1 voter, and that voter supports A. There is only one possible order of arrival, and A will be winning after the one and only vote.

Kickstart Practice Round 2017

A. Country Leader

B. Vote

C. Sherlock and Parentheses

Questions asked 1

Submissions

Country Leader

4pt Not attempted **366/497 users** correct (74%)

7pt Not attempted **279/360 users** correct (78%)

Vote

5pt Not attempted **227/304 users** correct (75%)

8pt Not attempted **165/214 users** correct (77%)

Sherlock and Parentheses

4pt Not attempted **257/277 users** correct (93%)

7pt Not attempted **220/256 users** correct (86%)

Top Scores

yashLadha	35
praran26	35
achaitanyasai	35
xhaler	35
iharsh234	35
Rajnikanth	35
sokokaleb	35
adtac	35
eon204	35
Irving.CL	35

Problem C. Sherlock and Parentheses

Small input
4 points — Solve C-small

Large input
7 points — Solve C-large

Problem

Sherlock and Watson have recently enrolled in a computer programming course. Today, the tutor taught them about the balanced parentheses problem. A string S consisting only of characters `(` and/or `)` is *balanced* if:

- It is the empty string, or:
- It has the form `(`S`)`, where S is a balanced string, or:
- It has the form S_1S_2, where S_1 is a balanced string and S_2 is a balanced string.

Sherlock coded up the solution very quickly and started bragging about how good he is, so Watson gave him a problem to test his knowledge. He asked Sherlock to generate a string S of **L** + **R** characters, in which there are a total of **L** left parentheses `(` and a total of **R** right parentheses `)`. Moreover, the string must have as many different balanced non-empty substrings as possible. (Two substrings are considered different as long as they start or end at different indexes of the string, even if their content happens to be the same). Note that S itself does not have to be balanced.

Sherlock is sure that once he knows the maximum possible number of balanced non-empty substrings, he will be able to solve the problem. Can you help him find that maximum number?

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: **L** and **R**.

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 answer, as described above.

Limits

$1 \le \mathbf{T} \le 100$.

Small dataset

$0 \le \mathbf{L} \le 20$.
$0 \le \mathbf{R} \le 20$.
$1 \le \mathbf{L} + \mathbf{R} \le 20$.

Large dataset

$0 \le \mathbf{L} \le 10^5$.
$0 \le \mathbf{R} \le 10^5$.
$1 \le \mathbf{L} + \mathbf{R} \le 10^5$.

Sample

Input	Output
3 1 0 1 1 3 2	Case #1: 0 Case #2: 1 Case #3: 3

In Case 1, the only possible string is `(`. There are no balanced non-empty substrings.
In Case 2, the optimal string is `()`. There is only one balanced non-empty substring: the entire string itself.
In Case 3, both strings `()()(` and `(()()` give the same optimal answer.
For the case `()()(`, for example, the three balanced substrings are `()` from indexes 1 to 2, `()` from indexes 3 to 4, and `()()` from indexes 1 to 4.