Challenge 15 - Seat change!

« Prev Next »

At today's unbirthday tea party, Alice notices that there's an unusual number of chairs at the table.
'What's the meaning of this?', she asks the Mad Hatter.
'Well, of course, we're going to have our regular unbirthday party.', answers the Mad Hatter.
'SEAT CHANGE!!!!' screams the Hare, suddenly.

And chaos ensues.
...

After all of them have settled down, the Hare screams 'SEAT CHANGE!!!!' and chaos ensues... again!

The Hare seems to be madder than usual today and 'SEAT CHANGE!!!' is all that can be heard throughout the morning. Alice, fed up with all the seat changes, decides to step aside and wait patiently.
After some time, Alice notices that the probability of reaching a specific seat from another one is constant and wonders where she'd have ended up, but, as she is still dizzy from all the seat changes, she asks you for help:
She will provide you with the probability of moving from several of the chairs to others and the number of times SEAT CHANGE!!! is shouted during the morning, and you need to tell her the seat in which she would have most likely have ended up.

'What about precision?', you ask Alice
'Precision? Precision! Precision is of the utmost importance. I want to know the last digit of both the numerator and the denominator of the resulting probability irreducible fraction!' the Mad Hatter interrupts, and Alice, tired of it all, just shrugs and nods.

Input

You will be provided with a definition of the party:
Firstly you get the number of Chairs at the table, C.
After that comes P, the number of given probabilities. All undefined probabilities are 0.
This is followed by P lines, each one composed of three integers: CHAIR_ORIGIN CHAIR_DEST PROBABILITY/100 (Probability given as percentage)
This is followed in turn by an integer, Q, the number of questions.
And finally Q lines, each with 2 integers: INITIAL_CHAIR NUMBER_OF_SEAT_CHANGES

Output

For each question, you need to calculate the chair with the highest probability, and output it in the format
"Case #Q: Chair: C Last digits: D/d"
In case of a tie, output the chair with the highest index, and remember that you only need to provide the last digits of the irreducible fraction.

Limits

$1 \leq C \leq 50$
$0 \leq CHAIR \leq C-1$
$0 \leq PROBABILITY \leq 100$
$1 \leq Q \leq 20$
$1 \leq NUMBER\_OF\_SEAT\_CHANGES \leq 100000000$

Sample Input

```
10
14
0 0 100/100
1 1 100/100
2 3 100/100
3 4 100/100
4 2 100/100
5 1 25/100
5 8 75/100
6 6 50/100
6 7 50/100
7 6 50/100
7 7 50/100
8 8 50/100
8 9 50/100
9 8 100/100
7
0 5
2 5
2 6
5 5
5 6
6 3
8 2
```

Sample Output

```
Case #1: Chair: 0 Last digits: 1/1
Case #2: Chair: 4 Last digits: 1/1
Case #3: Chair: 2 Last digits: 1/1
Case #4: Chair: 8 Last digits: 3/4
Case #5: Chair: 8 Last digits: 3/8
Case #6: Chair: 7 Last digits: 1/2
Case #7: Chair: 8 Last digits: 3/4
```

For example, in Case 7, on the first change you can reach either chair 8 or chair 9 each of which have a probability of 50/100.
On the second change, from chair 8 you can reach 8 or 9 again (50/100 * 50/100 and 50/100 * 50/100 respectively) and from chair 9 you are certain to go to chair 8

(50/100 * 100/100). This means that the probability of reaching chair 8 is 75/100, which rounds down to 3/4.

Test your code

You can test your program against both the input provided in the test phase and the input provided in the submit phase. A nice output will tell you if your program got the right solution or not. You can try as many times as you want to. Be careful with extra whitespaces, the output should be exactly as described.

Test your program against the input provided in the test phase

Download test input

Program output:

Seleccionar archivo Ningún archivo seleccionado

Submit test output

Test your program against the input provided in the submit phase

Download input

Program output:

Seleccionar archivo Ningún archivo seleccionado

Submit output

During the submit phase, in some problems, we might give your program harder inputs. As with the test token, a nice output will tell you if your program got the right solution or not. You can try as many times as you need.

In the actual contest you first need to solve the test phase before submitting the code, you must provide the source code used to solve the challenge and you can only submit once (once your solution is submitted you won't be able to amend it to fix issues or make it faster).

If you have any doubts, please check the info section.

« Prev Next »

Tweet about this! #TuentiChallenge6

Follow @TuentiEng