Coding Area

tcscodevita.com/main_page.jsp

Bus Travel

Problem Description

Amidst stiff competition the transport operators in the city are not in a position to increase the bus fare. One such public transport operator is you, who is suspecting some revenue loss despite the popularity of the bus service. Your reliable sources tipped you off that there are some passengers who take the advantage of overly crowded buses and take partial tickets (They take a ticket from a bus stop which is later than their actual origin; or they exit bus much later than their destination on their ticket). The in-out sensors you placed at the bus entry and exit gates provides you the exact details of the traveler's journey. Your objective is to find the bus stops from which these kinds of malpractices occur. You have to find out which passengers travelled with invalid tickets, and when they have over travelled (OT) or under travelled (UT).

A passenger is said to have over travelled if he does not have a valid ticket for all portions of his travel. Similarly, if a passenger has a valid ticket but alights the bus before his actual destination then he is said to have under travelled.

You have the data about every passengers' source and destination at which they boarded and alighted the bus respectively. You also have data about how many tickets were issued from which source to which destination. Using this information, you have to find out the passengers who OT or UT. For this purpose, you will have to adopt a ticket allocation policy which will map a ticket to a given passenger. Rules of ticket allocation policy in order of their importance are mentioned below. They have to be applied sequentially to do the ticket-to-passenger mapping.

1. For all tickets whose source and destination match with passenger data i.e., the stops at which they boarded and alighted should be allocated first. Passenger order is preserved for ticket allocation i.e. First matching ticket is allocated to the person who boarded first. Refer Example 1 in the *examples section* to get a better understanding.
2. For all tickets whose source matches with the source stop in the ticket data, try and do allocation if possible.
3. For all tickets whose destination matches with the destination stop in the ticket data, try and do allocation if possible.
4. If neither source nor destination data matches with any tickets data, then sequentially allocate the remaining tickets in the order of passenger boarding.

For better understanding, please refer to example section.

Note:

- None of the passenger can travel in the same bus again
- Assume none of the passenger will get ticket before boarding
- Assume passenger names are unique

Constraints

0 < N < 50

0 < T < 100

Input

First line contains an integer N denoting the number of bus stops in a bus journey.

Next N lines contain a string which provides passenger boarding and alighting information per stop. Format is as follows:

- There is an alphabetical string containing a pipe (|) character. The left-hand side of the pipe provides names of passenger boarding the bus and the right-hand side provides names of passengers alighting the bus.
- Passenger names are space delimited.
- When no passengers have boarded or alighted at a bus stop it would be denoted by (-).

Next line contains an integer T denoting the total number of tickets issued.

Next T lines contain two space delimited integers which denotes source and destination of each ticket respectively.

Output

Print the output in lexicographically sorted order of passenger names.

Each line of output should contain four things:

- First output will be passenger name
- Second output will be a number corresponding to start bus stop number
- Third output will be a number corresponding to end bus stop number
- Print "UT" if the travel type is under travel or "OT" if the travel type is over travel

OR

Print "VALID TRAVEL" when all the tickets are valid.

Time Limit (secs)

1

Examples

Example 1

Input

4

A B | -

C | -

- | A B

- | C

3

1 3

1 2

2 3

Output

B 2 3 OT

C 3 4 OT

Explanation

Here we have 4 bus stops and 3 passengers (A, B, C).

A and B have boarded from bus stop 1 and nobody has alighted at bus stop 1.

C has boarded from bus stop 2 and nobody has alighted at bus stop 2.

Nobody has boarded from bus stop 3 and A & B has alighted at bus stop 3.

Nobody has boarded from bus stop 4 and C has alighted at bus stop 4.

Total 3 tickets are issued.

First, A is allocated ticket #1 with source 1 and destination 3 because source and destination are matching.

Then, B is allocated ticket #2 with source 1 and destination 2 because only source is matching. B has alighted at stop 3. So, B has over travelled from 2 to 3.

Then, C is allocated ticket #3 with source 2 and destination 3 because only source is matching. C has alighted at stop 4. So, C has over travelled from 3 to 4.

Hence the output is printed in lexicographically sorted order of passenger name as depicted above.

Example 2

Input

5

A B R | -

C | B

D | A C

- | D

- | R

5

1 2

1 4

2 4

2 3

3 4

Output

A 3 4 UT

R 1 2 OT

R 4 5 OT

Explanation

Here we have 5 bus stops and 5 passengers (A, B, C, D, R).

A, B and R has boarded from bus stop 1 and nobody has alighted at bus stop 1.

C has boarded from bus stop 2 and B has alighted at bus stop 2.

D has boarded from bus stop 3 and A & C has alighted at bus stop 3.

Nobody has boarded from bus stop 4 and D has alighted at bus stop 4.

Nobody has boarded from bus stop 5 and R has alighted at bus stop 5.

Total 5 tickets are issued.

First, B is allocated ticket #1 with source 1 and destination 2 because source and destination are matching.

Then, C is allocated ticket #4 with source 2 and destination 3 because source and destination are matching.

Then, D is allocated ticket #5 with source 3 and destination 4 because source and destination are matching.

Then, A is allocated ticket #2 with source 1 and destination 4 because only source is matching. A has alighted at stop 3. So, A has under travelled from 3 to 4.

Now, R is allocated remaining ticket i.e., ticket #3 with source 2 and destination 4. R has boarded from bus stop 1 and alighted at bus stop 5. So, R has over travelled from 1 to 2 and 4 to 5.

Hence the output is printed in lexicographically sorted order of passenger name as depicted above.