3107 Joining Tables

A relational table of values may be used to represent a relation among data values wherein each row in the table represents a collection of related data values. Consider for example, the table (1)

PART0001	Glass Widget
PART0002	Steel Widget
PART0003	Porcelain Widget
PART0004	Wood Widget

From the first row of the table, "PART0001" is said to be related to "Glass Widget"; this row of related values is referred to as a tuple denoted by <"PART0001","Glass Widget">. The succeeding rows further indentify the following tuples: <"PART0002","Steel Widget">, <"PART0003","Porcelain Widget"> and <"PART0004","Wood Widget">

The EQUIJOIN operation is used to combine related tuples from two or more relational tables into single tuples. Equijoining tables involve performing a cross product on tuples that have identical values. Consider table (2):

PART0001	SALESMAN1
PART0002	SALESMAN2
PART0003	SALESMAN3
PART0004	SALESMAN4

We define an equijoin as VALID if and only if all rows from the original tables are complete and do not repeat. A valid equijoin of (1) and (2) would be on the "part" column:

PART0001	SALESMAN1	Glass Widget
PART0002	SALESMAN2	Steel Widget
PART0003	SALESMAN3	Porcelain Widget
PART0004	SALESMAN4	Wood Widget

For an invalid equijoin, consider a table (3):

Bird	Hawk
Mammal	Tiger
Fish	Shark
Reptile	Alligator

An equijoin between (3) and either of (1) and (2) is impossible because they have no common columns.

A partially correct equijoin would be of (1) and (2):

PART0001	SALESMAN1	Glass Widget
PART0002	SALESMAN2	Steel Widget
PART0003	SALESMAN3	Dog
PART0004	SALESMAN4	Wood Widget

In this case, the first two columns are said to be valid, but the third is invalid because it contains a value "Dog" which is not in the original values.

The problem is to determine whether the given set of relational tables, or any subset thereof, will produce valid equijoins.

Input

The first line in the input file indicates the number test cases.

Each test case begins with the number, t, of relational tables that will follow; where $2 \leq t \leq 16$. Each of the t relational tables is defined by a line containing the table's number of columns c and number of rows r separated by a single space; where $2 \leq c \leq 16$ and $1 \leq r \leq 64$. It is then followed by c sets of r data values of the relational table. Finally, a result table described in the above mentioned manner follows. All data values are strings and will be less than 80 characters.

Output

For each case, print the case number in one line, followed by c lines corresponding to number of columns of the given results table, containing either '`true`' or '`false`'. The line is '`true`' if and only if the column of the given results table belongs to a valid equijoin. If the entire table is invalid, such as when the equijoin is impossible, simply print '`false`' c number of times.

Sample Input

```
3
2
2 4
PART0001
PART0002
PART0003
PART0004
Glass Widget
Steel Widget
Porcelain Widget
Wood Widget
2 4
PART0001
PART0002
PART0003
PART0004
SALESMAN1
SALESMAN2
SALESMAN3
SALESMAN4
3 4
PART0001
PART0002
```

```
PART0003
PART0004
SALESMAN1
SALESMAN2
SALESMAN3
SALESMAN4
Glass Widget
Steel Widget
Porcelain Widget
Wood Widget
4
2 2
PART0001
PART0002
Glass Widget
Steel Widget
2 2
PART0001
PART0002
100
50
2 2
PART0001
PART0002
ShopOne
ShopTwo
2 2
ShopOne
ShopTwo
Mandaluyong
Pasig
2 2
PART0001
PART0002
Mandaluyong
Pasig
2
2 2
A1
A2
B1
B2
2 2
C1
C2
D1
D2
2 1
A1
C1
```

Sample Output

```
1
true
true
false

2
false
false

3
false
false
```