Problem Q

Can We Make a Reservation?

Problem ID: restaurants

Your solution to this problem must be implemented using an object-oriented approach by defining the classes presented below. Any other solution will receive zero points, even if the testing system judges it correctly.

Businesses often use software to manage certain aspects of their operations. For example, restaurants might use a reservation booking service like OpenTable or Resy. In this problem, you will define a `Reservation` class and a `Restaurant` class to implement a reservation system for a booking service.

To request a reservation a customer needs to provide: the name of the restaurant, the name of their party, and the number of guests in their party. Here is some sample reservation data:

Restaurant Name	Party Name	Number of Guests
Nella	Hammond	5
Medici	Bolton	5
Medici	Lumbergh	3
Nella	Malcolm	3
Medici	Smykowski	8
Nella	Grant	4
Medici	Joanna	9
Medici	Waddams	2

Given a list of restaurants and reservation requests, our reservation system will attempt to confirm a reservation with the restaurant. For every confirmed reservation, the restaurant needs to keep track of the party name, and the tables where the party will be seated. If a restaurant does not have enough tables to accommodate the reservation, a reservation is not made for that party. Instead, restaurants keep a tally of how many parties they are not able to seat. For this problem, you can make the following assumptions about restaurants:

- Each table has four seats.
- Each table is used only once (like at a dinner theater).
- Tables are numbered 0 through N - 1, where N is the number of tables.
- Parties can take up more than one table, but there can only be one party per table
- Parties are seated at empty tables in table order (i.e., starting from table 0 through N - 1).

To create your reservation system, you **must** implement the following class definitions along with the specified constructor, attributes (also known as instance variables) and methods. You must think about the type for each attribute and the type signature for each method. For this problem, the *Input* section helps clarify these types. Implement the following classes:

A class named `Reservation` that encapsulates information about a reservation.

- **Constructor**
  - Implement a constructor that initializes a `Reservation` instance and takes in two arguments. One argument is the name of the party for the reservation and the other is a list of the assigned tables for the reservation. For example, the reservation will be assigned to one or more tables from 0 to $N - 1$ for a specific restaurant.
- **Attribute(s)**
  - Define an attribute that represents the party name for the reservation.
  - Define an attribute that keeps track of the tables assigned to the reservation.
- **Method(s)**
  - No additional methods are required for this class.

A class named `Restaurant` for representing a restaurant.

- **Constructor**
  - Implement a constructor that initializes a `Resturant` instance and takes in two arguments. One argument is the name of the restaurant and the other is the number of tables available at the restaurant.
- **Attribute(s)**
  - Define an attribute that represents the name of the restaurant.
  - Define an attribute that represents the number of tables available at the restaurant.
  - Define an attribute that maintains the number of confirmed reservations. This attribute must hold `Reservation` objects.
  - Define an attribute that represents a count of the number of parties the restaurant was not able to accommodate.
- **Method(s)**
  - Implement a method called `make_reservation` in the `Restaurant` class that takes in two arguments. One argument is the party name for the reservation. The other argument is the number of guests for the party. This method will update the attributes of the restaurant based on trying to process and confirm the reservation request.

You may write additional helper methods and attributes; however, the above class definitions, constructor, attributes, and methods are **required** to be implemented and used in your solution.

As a remainder, your solution to this problem must be implemented using an object-oriented approach by defining the classes presented above. Any other solution will receive zero points, even if the testing system judges it correctly.

Input

The first input line is two integers separated by a single space; The first integer is N ($1 \leq N \leq 100$), which represents the number of restaurants in the booking service. The second integer is M ($1 \leq M \leq 100$), which represents the number of reservations in the service.

The input is followed by N lines, each corresponding to a single restaurant. A reservation line contains two components, each separated by a single space. The first component is the restaurant name and the second component is an integer T ($1 \leq T \leq 50$) that represents the total number of empty tables. Each resturant name contains only lowercase letters, no whitespace, and has a maximum length of 50 characters.

The remaining input is M lines, each corresponding to a single reservation. A reservation line contains three components, each separated by a single space. The first component is the restaurant name and the second component is the party name. Each party and restaurant name contains only lowercase letters, no whitespace, and has a maximum length of 50 characters. The third component is an integer G ($1 \leq G \leq 100$), which represents the number of guests in the party. You can assume the restaurant name is one from the N input lines.

Output

The output is the total number of parties not accommodated from all restaurants in the booking service and all confirmed reservations for each restaurant.

The first line is the total number of parties not accommodated from all restaurants in the booking service.

The remaining output lines are confirmed reservations from all restaurants in the service. The reservations **must** be in the same order they appeared in the input. Each reservation line contains the restaurant name, party name, followed by C integers, each line component is separated by a single space. The C integers are the assigned table numbers for the reservation.

Sample Input 1	**Sample Output 1**
2 8 nella 5 medici 8 nella hammond 5 medici bolton 5 medici lumbergh 3 nella malcolm 3 medici smykowski 8 nella grant 4 medici joanna 9 medici waddams 2	1 nella hammond 0 1 medici bolton 0 1 medici lumbergh 2 nella malcolm 2 medici smykowski 3 4 nella grant 3 medici joanna 5 6 7

Sample Input 2	Sample Output 2
2 3 valois 4 salonica 2 salonica patel 2 valois strayer 2 valois williams 9	0 salonica patel 0 valois strayer 0 valois williams 1 2 3