Árboles y Grafos, 2025-1

Para entregar el domingo 2 de marzo de 2025
A las 23:59 en la arena de programación

Instrucciones para la entrega

- Para esta tarea y todas las tareas futuras en la arena de programación, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada archivo de código (a modo de comentario). Adicionalmente, cite cualquier fuente de información que utilizó. Los códigos fuente que suba a la arena de programación deben der de su completa autoría.
- En cada problema debe leer los datos de entrada de la forma en la que se indica en el enunciado y debe imprimir los resultados con el formato allí indicado. No debe agregar mensajes ni agregar o eliminar datos en el proceso de lectura. La omisión de esta indicación puede generar que su programa no sea aceptado en la arena de programación.
- Puede resolver los ejercicios en C/C++ y Python. Sin embargo, deben haber por los menos soluciones a dos problemas en cada lenguaje.
- Debe enviar sus soluciones a través de la arena. Antes de subir sus soluciones asegurese de realizar pruebas con los casos de pruebas proporcionados para verificar que el programa es correcto, que finaliza y no se quede en un ciclo infinito.
- El primer criterío de evaluación será la aceptación del problema en la arena cumpliendo los requisitos indicados en los enunciados de los ejercicios y en este documento. El segundo criterio de evaluación será la complejidad computacional de la solución y el uso de los temas vistos en clase. Es necesario incluir en la cabecera del archivo comentarios que expliquen la complejidad de la solución del problema para cada caso.

  En adición a lo anterior, para efectos de la calificación se tendrán en cuenta aspectos de estilo como no usar `break` ni `continue` y que las funciones deben tener únicamente una instrucción `return` que debe estar en la última línea.

Problemas prácticos

Hay cuatro problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - Problem A

Source file name: `coaches.cpp`
Time limit: 1 second

Zlatan is a passionate football fan and greatly enjoys learning about the history of this sport, the most important in the world. He has made an effort to review books, newspapers, and videos about both past and present teams and players.

Zlatan keeps track of the coaches he knows and the players they have trained, and he has noticed something quite interesting. There are some coaches who may have never been players and, therefore, were never coached by anyone—or if they were, Zlatan has no record of any coach who trained them. He has labeled these coaches as the ”Generation 0 coaches.”

What really caught Zlatan’s attention is that some players, after becoming coaches, were only trained by Generation 0 coaches. He has called these coaches the ”Generation 1 coaches.” Zlatan has also observed that, following the same pattern, multiple generations of coaches can be identified.

Zlatan tends to obsess over statistics and would like to determine, for each generation of coaches, which coach has trained the most players. Since multiple coaches from the same generation may have the highest number of trained players, Zlatan will choose the oldest among them. If there is still a tie, the coach whose name is lexicographically smaller will be selected.

Input

There will be an integer T denoting the number of test cases, then, T test cases will follow. Each test case starts with two integers N ($1 \leq N \leq 2000$) and M ($1 \leq M \leq 250000$), the number of coaches and players in the record of Zlatan. The next R lines consists of a string of lowercase letter S and a number corresponding to the name and birth year of each coach or player. The following M lines will contain strings A and B, which indicate that A have trained B.

The input must be read from standard input.

Output

You will output for each test case the string `''Scenario #i:''` where `i` is the test case you are analyzing, after that, you will print a line for each generation and in each of them must be string `''Generation j:''` where `j` is the generation number and the name of the coach with the most trained players following the criteria described above.

The outputs of two consecutive cases will be separated by a blank line.

The output must be written to standard output.

Sample Input	**Sample Output**
1 12 17 Pirlo 1979 Cristiano-Ronaldo 1985 Capello 1946 Invernizzi 1931 Rocco 1912 Mouriño 1963 Xabi-Alonso 1981 Trapattoni 1939 Zlatan 1981 Wirtz 2003 Sacchi 1946 Ancelotti 1959 Capello Pirlo Pirlo Cristiano-Ronaldo Capello Cristiano-Ronaldo Mouriño Cristiano-Ronaldo Ancelotti Cristiano-Ronaldo Xabi-Alonso Wirtz Ancelotti Zlatan Invernizzi Trapattoni Rocco Trapattoni Invernizzi Sacchi Ancelotti Xabi-Alonso Mouriño Zlatan Capello Ancelotti Sacchi Ancelotti Trapattoni Ancelotti Trapattoni Sacchi Rocco Capello	Scenario #1: Generation 0: Rocco Generation 1: Capello Generation 2: Sacchi Generation 3: Ancelotti Generation 4: Xabi-Alonso

B - Problem B

Source file name: `forest.cpp`
Time limit: 2.5 seconds

Zlatan is very concerned about deforestation and bought a piece of land near Dapa with the purpose of planting trees and thus helping the planet. The land in question has a rectangular area and can be viewed as a grid with N rows and M columns. In each position of this grid, Zlatan planted trees over the years, trying to ensure that trees of multiple species were grouped together and he attempted to plant various species. After some time, the trees grew, and Zlatan's land now looks like a beautiful forest.

A biologist named Katia approached Zlatan to ask for his permission to explore the forest, which Zlatan agreed to, as he thought it was a good idea and was also very interested in Katia. Katia's goal is to determine which is the tallest tree in each grove planted by Zlatan. A grove corresponds to a region of trees of the same species that are adjacent either vertically or horizontally. In Zlatan's forest, there can be groves of the same species.

Your job is to help Katia determine the height of the tallest tree in each grove.

Input

The first line contains the number of test cases T. Each test case consists of a line with two numbers N and M ($2 \leq N, M \leq 1000$), which are the height and the width of the forest. Then follow N lines with M strings. Each one of these strings represent a tree and it is of the form `S#L` where `S` is the name of the tree specie and `L` is the height of the tree.

The input must be read from standard input.

Output

For each test case print `'Forest #n'`, where `n` is the number of the test case. After that, print the list of species of each grove and the height of the tallest tree in it. This list should be sorted by the species name lexicographically and by the height of the tallest tree when there are groves with the same species of trees.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 8 a#10 a#6 a#7 b#8 c#11 c#10 c#11 c#2 a#8 a#12 b#2 b#7 b#10 c#9 c#5 a#2 d#12 a#16 d#11 b#3 b#14 c#6 c#1 c#4 d#2 d#4 d#7 d#8 e#10 c#8 e#2 f#15 5 8 a#10 a#6 a#7 b#8 c#11 c#10 c#11 c#2 a#8 a#12 b#2 b#7 b#10 c#9 c#5 a#2 d#12 a#16 d#11 b#3 b#14 c#6 c#1 c#4 d#2 d#4 d#7 e#8 e#10 c#8 f#5 f#1 a#1 a#4 a#3 a#9 a#11 a#3 a#7 f#6	Forest #1 a 2 a 16 b 14 c 11 d 12 e 2 e 10 f 15 Forest #2 a 2 a 11 a 16 b 14 c 11 d 12 e 10 f 6

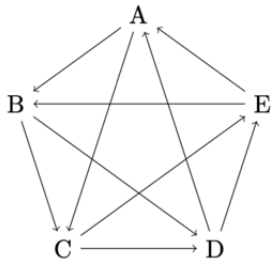
C - Problem C

Source file name: `money.cpp`
Time limit: 6 seconds

The International Commission for the Prevention of Cons is studying the possible effects of a pyramid scheme in a town. The scheme is as follows: someone asks a person for $1 and tells them to ask two other people for $1 each and to tell each of them to ask for money from two others just as they are doing. In this way, the victim thinks that they are going to earn $1. As there is a finite number of people in the world, not everyone can earn money this way, this is a con.

The N people in town are susceptible to the con, that is, they are willing to give $1 and later ask for money from two other people. However, they are willing to participate only once, that is, if they are asked for money again they will not give it or ask anyone. Once a person is asked for money, they give it immediately but can take some time before asking the other two people. The con starts with someone from outside the town asking someone in the town for money. This triggers a sequence of requests for money within the town.

For example, in the picture below we depict a town with five people. An arrow from A to B indicates that A would ask B for the money.



In this example, B can lose money. We can check that with the following scenario.

1. Someone from outside the town asks A for money.
2. A asks B for money.
3. A asks C for money.
4. C asks D for money.
5. B asks C for money.
6. B asks D for money.

Observe that when B asks C and D for money, they will not give it to B since they have already given money to someone else.

For each person in the town you know whom they are going to ask for money. Your task is to determine who in the town can lose money.

Input

The first line of each test case contains an integer N ($3 \leq N \leq 1000$) indicating the number of people in the town. Each person is identified by a distinct integer from 1 to N. For $i = 1, 2, \ldots, N$, the i-th of the next N lines contains two integers X_i and Y_i ($1 \leq X_i, Y_i \leq N$, $X_i, Y_i \neq i$ and $X_i \neq Y_i$), representing that person i would ask for money to person X_i and person Y_i.

The input must be read from standard input.

Output

For each test case output a single line with a string of length N such that its i-th character is the uppercase letter `'Y'` if person i can lose money, and the uppercase letter `'N'` otherwise.

The output must be written to standard output.

Sample Input	**Sample Output**
5 2 3 3 4 4 5 5 1 1 2 4 2 3 3 4 2 4 2 3	YYYYY NYYY

D - Problem D

Source file name: `shipping.cpp`
Time limit: 1 second

The Slow Boat to China Shipping company needs a program to help them quickly quote costs to prospective customers. The cost of a shipment depends on the size of the shipment and on how many shipping legs it requires. A shipping leg connects two warehouses, but since every pair of warehouses is not directly connected by a leg, it might require more than one leg to send a shipment from one warehouse to another.

A data set can represent from 1 to 300 warehouses. A two-letter code name will identify each warehouse (capital letters only). Shipping legs can exist between any two distinct warehouses. All legs are bidirectional.

The cost of a shipment is equal to the size of the shipment times the number of shipping legs required times \$100**.**

The input to the program identifies the warehouse code names and the existence of all shipping legs. For a given shipping request, consisting of the size of the shipment, the source warehouse and the destination warehouse, the program will output the best (cheapest) cost for the shipment, if it is possible to send shipments from the requested source to the requested destination. Alternately, the program must state that the request cannot be fulfilled.

Input

The first line will contain an integer that represents the number of data sets in the input. Each data set represents a new shipping configuration.

The first line of data in a data set will contain three integers, say M, N, and P: M is an integer ($1 \leq M \leq 300$) inclusive representing the number of warehouses in the data set; N is an integer from 0 to $M * (M - 1)/2$ inclusive that represents the number of legs between warehouses in the data set; P is an integer ($1 \leq P \leq 100$) inclusive that represents the number of shipping requests for which cost information is required.

The second line of data in a data set contains M two-letter code names for the M warehouses of the data sets. Only capital letters are used. A single blank separates code names. N lines follow the line of code names, containing shipping leg information in the format: `'XXX YYY'`, with `XXX` and `YYY` being the codes for two distinct warehouses in the set that have a direct link (a shipping leg) between them. There will be a single blank between the warehouse codes.

The N lines of shipping leg information are followed by P lines of shipping requests, one request per line. Each shipping request will begin with an integer that represents the size of the shipment. The shipment size will be followed by a pair of code names in the format `'AAA BBB'`, with `AAA` and `BBB` being the code for two distinct warehouses in the set which represent the source and destination of the requested shipment.

The input will be valid and consistent. A shipping leg will only be represented once within a data set. Data about legs will always refer to warehouses that have been identified as belonging to the data set. See the example below for clarification of the input format.

The input must be read from standard input.

Output

The first line of output should read `'SHIPPING ROUTES OUTPUT'`. For each data set there should be a section of output enumerating which data set the output section represents followed by P lines of the required information. Each of the P lines should list either the cheapest cost of the respective shipment or the phrase `'NO SHIPMENT POSSIBLE'`. The end of the output should be noted also. Produce output consistent with the example below.

The output must be written to standard output.

Sample Input	**Sample Output**
2 6 7 5 AAA CCC QQR FFF DDD AAB AAA CCC CCC QQR DDD CCC AAA DDD AAA AAB DDD QQR AAB DDD 5 AAA AAB 14 DDD CCC 1 CCC DDD 2 AAA FFF 13 AAB QQR 3 0 1 AAA BBB CCC 5 AAA CCC	SHIPPING ROUTES OUTPUT DATA SET 1 $500 $1400 $100 NO SHIPMENT POSSIBLE $2600 DATA SET 2 NO SHIPMENT POSSIBLE END OF OUTPUT