Árboles y Grafos, 2025-2

Para entregar el domingo 3 de agosto 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 finalice 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. Además, 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 cinco problemas prácticos cuyos enunciados aparecen a partir de la siguiente página. Debe resolver 4 de los 5 problemas. El problema restante se calificará con **0.5 décimas de bonificación**. En caso de que la nota total de la tarea supere 5.0 el excedente será puesto en otra tarea del curso.

A - Problem A

Source file name: `cargo.cpp`
Time limit: x seconds

The Plussians believe that the *Airport Based Cargo Distribution* & *Embarkation Facility* (*ABCDEF* in short) at the *Cosco International Airport* (*CIA*) is the largest such facility in the whole world though second largest in Plussia. Cargo planes arrive from and leave for many countries all around the world (Cermany, Q.S.A, Capan, Custralia etc. are some to name). *ABCDEF* is engaged in distributing, loading and unloading the cargoes carried by these planes.

Each cargo is a cubic box of fixed size and has a tag attached to it naming the destination country. For convenience, each country is assigned a unique ID. For example, if there are n countries, each country gets a unique integer ID ranging from 1 to n.

Every country X has its own cargo station identified by its country ID. There are two platforms (platform A and platform B) in each station. In platform A are put those cargoes which are to be transported (by air) to country X at some convenient time. Platform B is actually a queue of cargoes which are to be carried to countries other than country X. The cargo stations have a circular (ring) arrangement, that is, if there are n stations then each of the following station-pairs are adjacent: $(1, 2), (2, 3), (3, 4), \ldots, (n-1, n)$ and $(n, 1)$.

ABCDEF has many land-based cargo carriers used to carry cargo from station to station. But these carriers have so narrow space that you cannot even put two cargoes side by side. A carrier can carry more than one cargo (although there is a limit on the maximum number of cargoes a carrier can carry) only by putting them (the cargoes) on top of one another. This stack arrangement has the problem that you cannot just remove any cargo from the stack as you wish. For example, to remove the 3rd cargo from the top you must first remove the topmost two cargoes.

A cargo carrier moves from station to station strictly following their ring arrangement, that is, from station 1 it moves to station 2, then to 3, then 4, ..., then n, then 1 again, etc.. It requires exactly 2 minutes to move from any station to its adjacent one.

After reaching any station (say, station X), the cargo carrier first attempts to unload cargo. Starting from the topmost cargo in its stack, it checks the tag attached to the cargo. If it finds that the cargo has destination X, then it unloads it (the cargo) to platform A, otherwise it checks to see whether the queue in platform B has any vacant position, and if so it puts the cargo at the rear of the queue. This unloading procedure continues from the top to the bottom of the stack until one fails or the stack becomes empty, whichever comes first. Each successful unloading attempt requires exactly 1 minute, that is, unloading 3 cargoes in a station will require exactly 3 minutes. After unloading is complete, the carrier begins to load. The carrier continues to take the cargo in front of the queue in platform B and put it on top of its stack until the queue is empty or the stack is full, whichever comes first. Each successful loading attempt also requires exactly 1 minute, that is, loading 4 cargoes from a station will require exactly 4 minutes. After loading is complete, the carrier moves to the next station in the ring.

In this way, for many years, the cargo carriers are doing the job of moving the cargoes from platform B to platform A of appropriate stations from where the cargo planes carry them (the cargoes) to their destination countries.

But, after a conflict with the management regarding the pay scale, the employees of *ABCDEF* have gone to a strike for indefinite period from the last Sunday. Planes are arriving and leaving, but no one is there to load and unload the cargo. There is no one to distribute the queued cargoes to their destination stations. The entire facility has now come to a standstill.

But, you, as always known to and hated by your colleagues as the boss's man, have decided to break the strike and save *CIA*. You are going to start working from tomorrow morning and distribute the queued cargoes to their appropriate stations using a cargo carrier. Initially your carrier will be empty and, you will start your journey from station 1 and continue to move around the ring until all the cargoes have been distributed to their destination stations. But before starting the job, you have decided to write a program to determine exactly how long it will take to complete it.

Input

The input contains several sets of input. The first line of the input file contains an integer *SET*, which indicates how many sets of inputs are there. It is then followed by *SET* sets of inputs. In our sample input the value of *SET* is 2.

The first line of the input contains three integers: N, S and Q. N ($2 \leq N \leq 100$) is the number of stations in the ring. S ($1 \leq S \leq 100$) is the capacity of your cargo carrier, that is, the maximum number of cargoes your carrier can carry. Q ($1 \leq Q \leq 100$) is the maximum number of cargoes the queue in platform B can accommodate. All the queues in the system are assumed to have the same capacity.

Then follow N lines. Assuming that these lines are numbered from 1 to N, for $1 \leq i \leq N$, line i contains an integer Q_i ($0 \leq Q_i \leq Q$) giving the number of cargoes queued at station i followed by Q_i integers giving the destination stations of the queued cargoes from front to rear. You may assume that none of these Q_i cargoes will have station i as their destination.

The input must be read from standard input.

Output

For each set of input output the number of minutes it will take to finish the job in a separate line

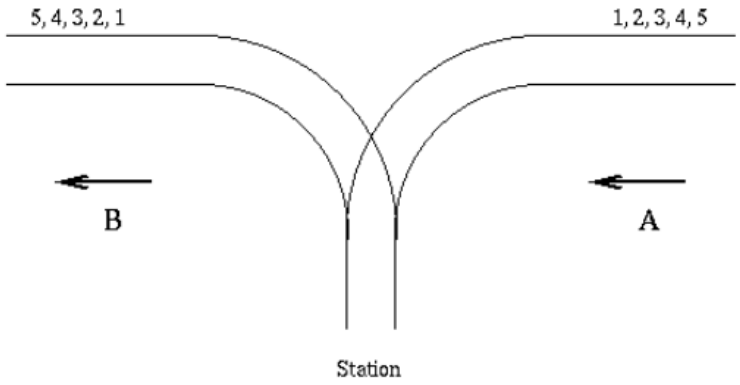
The output must be written to standard output.

Sample Input	**Sample Output**
2 5 2 3 3 4 5 2 2 1 3 0 3 3 5 1 1 4 5 2 3 3 4 5 2 2 1 3 0 3 3 5 1 1 4	72 72

B - Problem B

Source file name: `rails.cpp`
Time limit: x seconds

There is a famous railway station in PopPush City. Country there is incredibly hilly. The station was built in last century. Unfortunately, funds were extremely limited that time. It was possible to establish only a surface track. Moreover, it turned out that the station could be only a dead-end one (see picture) and due to lack of available space it could have only one track.



The local tradition is that every train arriving from the direction A continues in the direction B with coaches reorganized in some way. Assume that the train arriving from the direction A has $N \leq 1000$ coaches numbered in increasing order $1, 2, \ldots, N$. The chief for train reorganizations must know whether it is possible to marshal coaches continuing in the direction B so that their order will be $a_1, a_2, \ldots, a_N$. Help him and write a program that decides whether it is possible to get the required order of coaches. You can assume that single coaches can be disconnected from the train before they enter the station and that they can move themselves until they are on the track in the direction B. You can also suppose that at any time there can be located as many coaches as necessary in the station. But once a coach has entered the station it cannot return to the track in the direction A and also once it has left the station in the direction B it cannot return back to the station.

Input

The input file consists of blocks of lines. Each block except the last describes one train and possibly more requirements for its reorganization. In the first line of the block there is the integer N described above. In each of the next lines of the block there is a permutation of $1, 2, \ldots, N$. The last line of the block contains just `'0'`.

The last block consists of just one line containing `'0'`.

The input must be read from standard input.

Output

The output file contains the lines corresponding to the lines with permutations in the input file. A line of the output file contains `'Yes'` if it is possible to marshal the coaches in the order required on the corresponding line of the input file. Otherwise it contains `'No'`. In addition, there is one empty line after the lines corresponding to one block of the input file. There is no line in the output file corresponding to the last "null" block of the input file.

The output must be written to standard output.

Sample Input	**Sample Output**
5 1 2 3 4 5 5 4 1 2 3 0 6 6 5 4 3 2 1 0 0	Yes No Yes

C - Problem C

Source file name: `snowflakes.cpp`
Time limit: x seconds

Emily the entrepreneur has a cool business idea: packaging and selling snowflakes. She has devised a machine that captures snowflakes as they fall, and serializes them into a stream of snowflakes that flow, one by one, into a package. Once the package is full, it is closed and shipped to be sold.

The marketing motto for the company is "bags of uniqueness." To live up to the motto, every snowflake in a package must be different from the others. Unfortunately, this is easier said than done, because in reality, many of the snowflakes flowing through the machine are identical. Emily would like to know the size of the largest possible package of unique snowflakes that can be created. The machine can start filling the package at any time, but once it starts, all snowflakes flowing from the machine must go into the package until the package is completed and sealed. The package can be completed and sealed before all of the snowflakes have flowed out of the machine.

Input

The first line of input contains one integer specifying the number of test cases to follow. Each test case begins with a line containing an integer n, the number of snowflakes processed by the machine. The following n lines each contain an integer (in the range 0 to 10^9, inclusive) uniquely identifying a snowflake. Two snowflakes are identified by the same integer if and only if they are identical.

The input will contain no more than one million total snowflakes.

The input must be read from standard input.

Output

For each test case output a line containing single integer, the maximum number of unique snowflakes that can be in a package.

The output must be written to standard output.

Sample Input	**Sample Output**
1 5 1 2 3 2 1	3

D - Problem D

Source file name: `soccer.cpp`
Time limit: x seconds

A sports journalist has gained access to the results of a soccer league and wants to calculate the final standings. In this league, three points are given to the team winning a match, one point for each team in a draw, and none for a defeated team.

The standing of each team in the league shall be determined as follows:

1. greatest number of points obtained in all matches;
2. greatest goal difference in all matches (i.e., goals scored minus goals against);
3. greatest number of goals scored in all matches; and
4. greatest number of goals scored playing as visitor in all matches.

If two or more teams are equal on the basis of the above criteria, their rankings shall be determined by lexicographic order on the team's name (characters are sorted by ASCII value).

The journalist also wants to determine the number of occurrences of the classic sports journalist's paradox, namely, the number of matches in which the team losing the game has a better final standing than the one winning that game.

Input

The input consists of several test cases. The first line of each test case contains a natural number M indicating the number of matches ($1 \leq M \leq 64$). Each one of the next M lines contains the results of the matches in the format:

LX `vs.` YV

where X is the number of goals scored by the local team with name L ($0 \leq X \leq 32$, $1 \leq |L| \leq 100$) and Y is the number of goals scored by the visitor team with name V ($0 \leq Y \leq 32$, $1 \leq |V| \leq 100$, $V \neq L$).

You can assume that each team name consists of uppercase characters `'A'..'Z'`, digits `'0'..'9'`, periods (`.`), and can contain blanks. However, blanks do not appear at the beginning or end of a name.

The input must be read from standard input.

Output

For each test case, print a line with the text

`The paradox occurs` X `time(s).`

where X is the number of paradoxes found by the end of the league. This line should be followed by the final standings in the format

`1.` $Name_1$

`2.` $Name_2$

...

`N.` $Name_N$

where N is the number of teams in the league and such that the i-th place in the final standings is occupied by the team with name $Name_i$.

The output must be written to standard output.

Sample Input	**Sample Output**
13 B. DORTMUND 2 vs. 2 REAL MADRID SP. PORTUGAL 2 vs. 0 LEGIA SP. PORTUGAL 1 vs. 2 B. DORTMUND REAL MADRID 5 vs. 1 LEGIA B. DORTMUND 1 vs. 0 SP. PORTUGAL LEGIA 3 vs. 3 REAL MADRID MONACO 3 vs. 0 CSKA M. SP. PORTUGAL 1 vs. 2 REAL MADRID B. DORTMUND 8 vs. 4 LEGIA REAL MADRID 2 vs. 2 B. DORTMUND LEGIA 1 vs. 0 SP. PORTUGAL MONACO 1 vs. 0 SP. PORTUGAL CSKA M. 1 vs. 0 B. DORTMUND 2 TEAM 1 4 vs. 2 TEAM 2 TEAM 2 2 vs. 0 TEAM 1	The paradox occurs 2 time(s). 1. B. DORTMUND 2. REAL MADRID 3. MONACO 4. LEGIA 5. CSKA M. 6. SP. PORTUGAL The paradox occurs 1 time(s). 1. TEAM 2 2. TEAM 1

E - Problem E

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

Zlatan is an avid collector of various items. Among the things Zlatan enjoys collecting are scale model cars, figures from animated series, and cards from Jet chocolate albums. To obtain the album cards, Zlatan must buy chocolates, each containing one card. Albums typically require a large number of cards, meaning Zlatan has to buy and consume many chocolates. However, Zlatan has had issues with sugar consumption and prefers to avoid such foods. Consequently, Zlatan buys many chocolates to give away, and he takes advantage of this to earn points with women he courts, as it is said that one way to a woman's heart is through gifting chocolates.

Over time, Zlatan has managed to collect a significant number of album cards, but it has become increasingly difficult to find the missing ones since most of the cards he gets are duplicates. As a result, Zlatan has accumulated a large number of duplicate cards. Teofilo, his best friend, is willing to help him get rid of some of these duplicates and has offered to buy one unit of the k most duplicated cards Zlatan owns. Zlatan, however, does not have a record of which cards or how many times each card appears in his collection. For each card, Zlatan only knows its number and a label. Given Zlatan's collection, it is necessary to determine the k most duplicated cards.

Input

There are several cases in the input. First line of each test case contains numbers n ($n \leq 10^9$) and k ($k \leq min(n, 10^6)$) — the number of cards in the collection of Zlatan and the number of cards that Teofilo can buy. Then, there are n lines, each line contains a string corresponding to the label of the card and the serial number of the card. You can assume the label will have no spaces. The end of the input is indicated with two zeros for n and k.

The input must be read from standard input.

Output

For each test case, k lines must be displayed — the k most duplicated cards. Each line has the label of the card and the number of ocurrences for the card. If two cards have the same number of occurrences then the card with the greater serial number must be selected. The cards must be displayed in the lexicographical order.

The output must be written to standard output.

Sample Input	**Sample Output**
13 3 Wild-Dog 8 Sad-Cat 4 Wild-Dog 8 Sad-Cat 4 Silvester-Duck 5 Sweet-Worm 11 Happy-Penguin 6 Wise-Spider 2 Silvester-Duck 5 Fat-Monkey 3 Wild-Dog 8 Old-Dinasour 1 Sweet-Worm 11 0 0	Silvester-Duck 2 Sweet-Worm 2 Wild-Dog 3