Árboles y Grafos, 2025-2

Para entregar el jueves 2 de octubre 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: `capital.cpp, capital.py`
Time limit: 2 seconds

There are N cities in Flatland connected with M unidirectional roads. The cities are numbered from 1 to N. The Flat Circle of Flatland (FCF) wants to set up a new capital city for his kingdom. For security reasons, *the capital must be reachable from all other cities* of Flatland. FCF needs the list of *all* candidate cities. You are the chief programmer at FACM (Flat Association for Computing Machinery) responsible for providing the list to FCF as soon as possible.

Input

The first line of each test case contains two integers: $1 \leq N \leq 100000$ and $1 \leq M \leq 200000$. Each of the following M lines contains two integers $1 \leq A, B \leq N$ denoting a road from A to B.

The input must be read from standard input.

Output

The output for each test case contains an integer denoting the number of candidate cities followed by the list of candidate cities in increasing order.

The output must be written to standard output.

Sample Input	**Sample Output**
4 4 1 2 3 2 4 3 2 1	2 1 2

B - Problem B

Source file name: `critical2.cpp, critical2.py`
Time limit: 4 seconds

In the Amicable, Great, Respected, and Advanced Empire of Zlatan, known as the AGRA Empire, train stations are connected by tracks where trains travel in both directions. Cities correspond to groups of stations that are interconnected, meaning that from any station, one can reach any other. There is a direct or indirect path between any pair of stations in the empire.

Some stations are extremely important because if they were unavailable, the stations in the cities could become disconnected. These stations are called *critical* stations. For this reason, Zlatan wants to avoid inconvenients with critical stations, but ensuring their security can be very expensive, and unfortunately, the empire's treasury is not in the best condition at the moment. Consequently, Zlatan has decided to establish a team dedicated to handling situations that may threaten the safety of critical stations. This team will have at its disposal a truck equipped with state-of-the-art technology to deal with all types of disasters. The problem is that there will be only one truck in the empire, so the team can attend to only one critical station at a time. Assuming that the truck takes t minutes to travel between any two stations that are directly connected, Zlatan would like to know what is the maximum amount of time it could take the truck to travel between two critical stations.

Input

There will be multiple test cases in the input. For each test case, there will be a line with two numbers, n and m ($1 \leq n \leq 3000, 1 \leq m \leq 50000$), corresponding to the number of stations and the number of connections between stations in the AGRA Empire. The next line contains the value of t ($0 < t \leq 10^5$). Then, there will be m lines, each containing two numbers, u and v ($0 \leq n, m < n$), indicating that there is a track between station u and station v.

Last case will be indicated with $n = m = 0$. This case should not be processed.

The input must be read from standard input.

Output

For each test case, if there are two or more critical stations in the empire then print a line with the maximum amount of time necessary to the truck travels between any two critical stations. Otherwise, the answer will be `0`.

The output must be written to standard output.

Sample Input

```
16 21
100
1 2
1 3
2 3
1 6
6 3
6 7
7 2
3 10
13 10
10 4
9 4
9 10
5 9
5 12
11 15
8 12
11 12
11 8
8 5
0 13
14 13
0 0
```

Sample Output

```
500
```

C - Problem C

Source file name: `mazes.cpp, mazes.py`
Time limit: 1 seconds

The Queen of Nlogonia is a fan of mazes, and therefore the queendom's architects built several mazes around the Queen's palace. Every maze built for the Queen is made of rooms connected by corridors. Each corridor connects a different pair of distinct rooms and can be transversed in both directions.

The Queen loves to stroll through a maze's rooms and corridors in the late afternoon. Her servants choose a different challenge for every day, that consists of finding a simple path from a start room to an end room in a maze. A simple path is a sequence of distinct rooms such that each pair of consecutive rooms in the sequence is connected by a corridor. In this case the first room of the sequence must be the start room, and the last room of the sequence must be the end room. The Queen thinks that a challenge is good when, among the routes from the start room to the end room, exactly one of them is a simple path. Can you help the Queen's servants to choose a challenge that pleases the Queen?

For doing so, write a program that given the description of a maze and a list of queries defining the start and end rooms, determines for each query whether that choice of rooms is a good challenge or not.

Input

Each test case is described using several lines. The first line contains three integers R, C and Q representing respectively the number of rooms in a maze ($2 \leq R \leq 10^4$), the number of corridors ($1 \leq C \leq 10^5$), and the number of queries ($1 \leq Q \leq 1000$). Rooms are identified by different integers from 1 to R. Each of the next C lines describes a corridor using two distinct integers A and B, indicating that there is a corridor connecting rooms A and B ($1 \leq A < B \leq R$). After that, each of the next Q lines describes a query using two distinct integers S and T indicating respectively the start and end rooms of a challenge ($1 \leq S < T \leq R$). You may assume that within each test case there is at most one corridor connecting each pair of rooms, and no two queries are the same.

The last test case is followed by a line containing three zeros.

The input must be read from standard input.

Output

For each test case output $Q + 1$ lines. In the i-th line write the answer to the i-th query. If the rooms make a good challenge, then write the character '`Y`' (uppercase). Otherwise write the character '`N`' (uppercase). Print a line containing a single character '`-`' (hyphen) after each test case.

The output must be written to standard output.

Sample Input	**Sample Output**
6 5 3 1 2 2 3 2 4 2 5 4 5 1 3 1 5 2 6 4 2 3 1 2 2 3 1 4 1 3 1 2 0 0 0	Y N N - N Y Y

D - Problem D

Source file name: `south-park.cpp, south-park.py`
Time limit: 1 seconds

At South Park Elementary School, Mr. Herbert Garrison has a difficult class to which he recently gave an exam. After analyzing the students' answers and questioning some of them, Mr. Garrison discovered that multiple students shared answers during the exam.

More specifically, he was able to gather information in the form of pairs (a, b) of students, where student a gave a copy of an answer to student b. From this information, he realized that there were several cheating clusters during the exam.

Peter Charles, the school principal, better known as Principal PC, is determined to sanction the students involved. However, due to school regulations, he must take into account the following:

- It is only possible to sanction groups of students in which it can be determined with certainty who initiated the cheating. Therefore, it is necessary to order the students according to their responsibility in the offense. If student a gave an answer to student b, then a has more responsibility than b, since a is accountable for propagating their answer further within the group. This responsibility relation is transitive.
- There is a special rule: if Eric Cartman is involved in any cheating cluster, then he must always be considered the most responsible student, regardless of any other factor.
- When it is not possible to establish an order of responsibility among the students involved in a particular cluster, those students must be excluded from the final list of sanctioned students.
- In addition, when two students could otherwise be considered equally responsible or two students are in different clusters, the following tie-breaking criteria apply, in order:
    1. The student who directly gave copies to more classmates is considered more responsible.
    2. If a tie still remains, the student who received copies from more classmates is considered more responsible.
    3. If there is still a tie, the student whose name is lexicographically greater is considered more responsible.

Principal PC has spared no expense and has hired the best programmer in town—that is, you—to determine the responsibility order of the students involved, based on the information collected by Mr. Garrison about who gave answers to whom.

Input

The input begins with an integer n indicating the number of reported copying relations. Each of the following n lines contains two strings a and b, meaning that student a gave an answer to student b. Each student's name consists of at most 20 uppercase and lowercase letters.

The input ends with a line containing a single 0.

The input must be read from standard input.

Output

Print in a single line the list of sanctioned students, ordered according to the rules described above. Students for whom it is not possible to establish a clear responsibility must not appear in the output.

The output must be written to standard output.

Sample Input

```
5
Stan Kyle
Kyle Kenny
Cartman Stan
Kenny Wendy
Wendy Bebe
6
Stan Kyle
Kyle Kenny
Wendy Bebe
Bebe Wendy
Cartman Clyde
Clyde Stan
9
Stan Kyle
Kyle Kenny
Wendy Bebe
Bebe Wendy
Butters Cartman
Butters Timmy
Timmy Kenny
Cartman Clyde
Clyde Stan
0
```

Sample Output

```
Cartman Stan Kyle Kenny
Cartman Clyde Stan Kyle Kenny
Cartman Butters Timmy Clyde Stan Kyle Kenny
```