Árboles y Grafos, 2025-1

Para entregar el domingo 2 de febrero 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: `10-20-30.cpp`
Time limit: 1 second

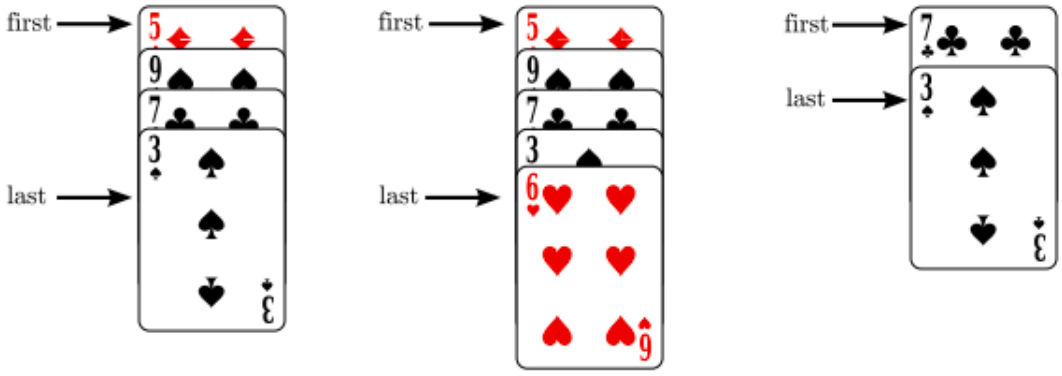
A simple solitaire card game called 10-20-30 uses a standard deck of 52 playing cards in which suit is irrelevant. The value of a face card (king, queen, jack) is 10. The value of an ace is one. The value of each of the other cards is the face value of the card (2, 3, 4, etc.). Cards are dealt from the top of the deck. You begin by dealing out seven cards, left to right forming seven piles. After playing a card on the rightmost pile, the next pile upon which you play a card is the leftmost pile.

For each card placed on a pile, check that pile to see if one of the following three card combinations totals 10, 20, or 30:

1. the first two and last one,
2. the first one and the last two, or
3. the last three cards.

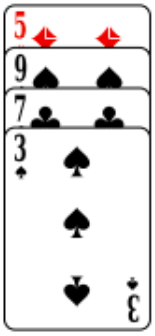
If so, pick up the three cards and place them on the bottom of the deck. For this problem, always check the pile in the order just described. Collect the cards in the order they appear on the pile and put them at the bottom of the deck. Picking up three cards may expose three more cards that can be picked up. If so, pick them up. Continue until no more sets of three can be picked up from the pile.

For example, suppose a pile contains 5 9 7 3 where the 5 is at the first card of the pile, and then a 6 is played. The first two cards plus the last card (5 + 9 + 6) sum to 20. The new contents of the pile after picking up those three cards becomes 7 3. Also, the bottommost card in the deck is now the 6, the card above it is the 9, and the one above the 9 is the 5.



original pile after playing 6 after picking up

If a queen were played instead of the six, 5 + 9 + 10 = 24, and 5 + 3 + 10 = 18, but 7 + 3 + 10 = 20, so the last three cards would be picked up, leaving the pile as 5 9.



original pile

after playing queen

after picking up

If a pile contains only three cards when the three sum to 10, 20, or 30, then the pile "disappears" when the cards are picked up. That is, subsequent play skips over the position that the now-empty pile occupied. You win if all the piles disappear. You lose if you are unable to deal a card. It is also possible to have a draw if neither of the previous two conditions ever occurs.

Write a program that will play games of 10-20-30 given initial card decks as input.

Input

Each input set consists of a sequence of 52 integers separated by spaces and/or ends of line. The integers represent card values of the initial deck for that game. The first integer is the top card of the deck. Input is terminated by a single zero (0) following the last deck.

The input must be read from standard input.

Output

For each input set, print whether the result of the game is a win, loss, or a draw, and print the number of times a card is dealt before the game results can be determined. (A draw occurs as soon as the state of the game is repeated.) Use the format shown in the "Sample Output" section.

The output must be written to standard output.

Sample Input	**Sample Output**
2 6 5 10 10 4 10 10 10 4 5 10 4 5 10 9 7 6 1 7 6 9 5 3 10 10 4 10 9 2 1 10 1 10 10 10 3 10 9 8 10 8 7 1 2 8 6 7 3 3 8 2 4 3 2 10 8 10 6 8 9 5 8 10 5 3 5 4 6 9 9 1 7 6 3 5 10 10 8 10 9 10 10 7 2 6 10 10 4 10 1 3 10 1 1 10 2 2 10 4 10 7 7 10 10 5 4 3 5 7 10 8 2 3 9 10 8 4 5 1 7 6 7 2 6 9 10 2 3 10 3 4 4 9 10 1 1 10 5 10 10 1 8 10 7 8 10 6 10 10 10 9 6 2 10 10 0	Win : 66 Loss: 82 Draw: 73

B - Problem B

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

Well, 5 of the trouble makers are going to Dinajpur by the well known train Drutojan Express. The 5 trouble makers — Ja, Sam, Sha, Sid and Tan were discussing about how to create trouble in Dinajpur. But after a while they themselves got into trouble. Well, let me share the story with you.

The tickets they bought were the cause of the problem. See, 4 of those tickets all shared the same compartment while the other ticket was for a different compartment. As you can guess, no one was willing to go to the compartment where the single seat is.

After almost two hours of quarrel and unmentionable fights, they were exhausted; so they came up with a plan. The plan is: each of them will make a list containing names of the other persons. Now, if a person goes to the single seat, first he will stay there for M minutes. After that he will exchange his seat with the person who is on top of his own list. At this point he will also update his list by removing this name from top of his list and add the name at the bottom of the list.

This will go on all through their journey. So you are given their lists, the person who is on the single seat right now and the total time N (in minutes) for which they would be traveling. For each of them, you have to find the total time he will stay in the single seat. You can assume that its takes 2 minutes to exchange seats. And you will start time counting from the 0-th minute. That means if M is 5 and the first person who is in the single is Sid, then he will exchange after 5 minutes. Then from the 7-th minute the next person in his list (Sha) will be in that seat. Remember that a person can leave the seat without staying M minutes if the train reaches the destination within M.

Input

The first line of the input will contain an integer T denoting the number of cases.

Each case starts with two integers M ($2 \leq M \leq 30$) and N ($10 \leq N \leq 1000$) and the name of the person who is in the single seat. Then there will be 5 lines. Each of the lines represents a list. A list contains and integer k ($1 \leq k \leq 20$) and k names separated by spaces. That means this list has k members and the names are listed from top to bottom. These 5 lines contain lists of Ja, Sam, Sha, Sid and Tan respectively. In a list, a name can occur more than once. You can assume no one will put his own name in his list.

The input must be read from standard input.

Output

For each case print the the case number in a single line. Then print 5 lines for each person (in the order shown below) and total time he will be on the single seat. Check samples for details. Report the names as in input.

Explanation:

- First 3 minutes Ja was on the single seat
- After that he exchanged his seat with Tan, it took 2 minutes
- After that Tan was there for 3 minutes.
- He exchanged with Ja again, and it took 2 minutes
- Ja stayed for 1 minute and they reached the destination. So, Ja was there for 4 minutes and Tan was there for 3 minutes.

The output must be written to standard output.

Sample Input	**Sample Output**
1 3 11 Ja 5 Tan Sid Sam Sha Tan 1 Ja 1 Ja 1 Ja 1 Ja	Case 1: Ja 4 Sam 0 Sha 0 Sid 0 Tan 3

C - Problem C

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

Some of you may have heard the joke about a boy who stopped going to school because he thought that his teacher was crazy. When his parents asked as to why he felt that way, the boy replied, "two days back, my math teacher said, $12 = 4 \times 3$, and yesterday she said $12 = 6 \times 2$. How can I trust a teacher like this who contradicts her own saying?". His parents explained the situation to him, so the boy was convinced and continued with his studies. Few years later, the boy wonders, how many ways are there to factorize a number. He sits with pen and paper and discovers that the task is getting quite tedious for large numbers. So he asks you for help. He wants you to write a program to solve his problem.

Input

Each line of input contains a positive integer, $N \leq 2000000$. The last case is followed by a value of 0 for N, which should not be processed.

The input must be read from standard input.

Output

For every case, there will be one or more lines of output. The first line of each set of output should contain the number F, where F is the number of ways to uniquely factorize N. If $F > 0$, then the next F lines should contain the factorizations of N. The numbers in the factorizations should be sorted in non-decreasing order and individual factorizations should be sorted in dictionary order. When printing numbers in a line, there should be a space between each number. See sample output of $N = 20$ for clarification. Permutations of factorization are considered same. That is, $12 = 4 \times 3$ is same as $12 = 3 \times 4$.

Note: To limit the answers to finite values, the factorizations should not contain any 1, and there should be at least two numbers in the factorizations. So for this problem, $20 = 1 \times 20$, is not a factorization of 20.

The output must be written to standard output.

Sample Input	**Sample Output**
1 20 0	0 3 2 2 5 2 10 4 5

D - Problem D

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

Cakey McCakeFace's signature pastry, the Unknowable Cake, is baked daily in their Paris facility. The make-or-break trick for this cake is the cooking time, which is a very well-kept secret. Eve, the well-known spy, wants to steal this secret, and your job is to help her.

Cakes are cooked in a single huge oven that has exactly one front and one back door. The uncooked cakes are inserted through the front door. After the exact and very secret cooking time has passed, the cakes exit the oven through the back door. Only one cake can go through the front or back door at any given time.

Eve has secretly installed detectors at the front and back of the oven. They record a signal every time a cake passes through the doors. A cake will therefore trigger the entry detector at some time t when it goes through the front door, and then trigger the exit detector at time exactly t + *cooking_time* when it goes through the back door (all cakes at Cakey McCakeFace are always perfectly cooked).

After a few days, she receives two sets of timestamps (in ms) corresponding to entry and exit detectors. Unfortunately, the detectors are faulty: they are sometimes triggered when no cake has passed, or they may fail to be triggered when a cake passes. Eve noticed that she could make a good guess of the secret *cooking_time* by finding the time difference that maximizes the number of correspondences of entry and exit detection times. Help Eve compute this.

Input

The input file contains several test cases, each of them as described below.

- **Line 1**: the number N of times the entry detector was triggered.
- **Line 2**: the number M of times the exit detector was triggered.
- **Line 3**: the N integer timestamps at which the entry detector was triggered, sorted in ascending order, with no repetition, space-separated.
- **Line 4**: the M integer timestamps at which the exit detector was triggered, sorted in ascending order, with no repetition, space-separated.

Limits

- $1 \leq N, M \leq 2000$;
- each timestamp is between 0 and 1000000000 (inclusive).

The input must be read from standard input.

Output

For each test case, the output must follow the description below.

A single integer: your best guess of the secret *cooking_time*, the (positive or zero) time difference that maximizes the number of correspondences of entry and exit detection times. If multiple such values exist, the smallest one should be returned.

The output must be written to standard output.

Sample Input	**Sample Output**
5 5 0 10 12 20 30 1 5 17 27 50	5

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 numers 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