

Concurso de Programación

AdaByron 2023

Final nacional



Cuadernillo de problemas

Patrocinado por



Realizado en la **Facultad de Informática (UCM)**
7-8 de julio de 2023



Facultad
de
Informática



Universidad
Complutense
Madrid

In almost every computation a great variety of arrangements for the succession of the processes is possible, and various considerations must influence the selections amongst them for the purposes of a calculating engine. One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.

Ada Byron

Listado de problemas

A. El juego de las puertas	3
B. ¡Hoy toca pisci!	5
C. Juez amable	7
D. Tres en raya	9
E. Pintoresca vieja Europa	11
F. Preparándose para hibernar	13
G. Willy	15
H. Lápices de colores	17
I. Miguelín Duráin	19
J. Jornada partida	21
K. Campamento de verano	23
L. ¡Vamos a la feria!	25

Autores y revisores de los problemas:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)
- Daniel Padilla Rodríguez (Universidad Complutense de Madrid)
- Pablo Hidalgo Palencia (Universidad Complutense de Madrid)
- Sofía Prieto Ibáñez (Universidad Complutense de Madrid)
- Alberto Maurel Serrano (Universidad Complutense de Madrid)
- Manuel Montenegro Montes (Universidad Complutense de Madrid)
- Isabel Pita Andreu (Universidad Complutense de Madrid)
- Javier del Río López (Universidad Complutense de Madrid)
- Rubén Rubio (Universidad Complutense de Madrid)
- Alberto Verdejo (Universidad Complutense de Madrid)

● A El juego de las puertas

En el clásico programa de televisión *El castillo de Takeshi* existe un famoso juego denominado ‘Knock Knock’, en el que los participantes se encuentran ante una serie de muros con 4 puertas de papel en cada uno.

Los participantes deben coger carrerilla y lanzarse contra una de las puertas para poder atravesar el muro y pasar al siguiente, donde deberán repetir la hazaña hasta llegar al final. Todas las puertas del mismo muro conducen al mismo sitio, pero el truco está en que muchas de las puertas son falsas (están pintadas sobre el muro), por lo que no pueden atravesarse, y los participantes deben tratar de identificarlas para no ser eliminados del juego y muy probablemente ir directos al hospital.



Inspirándose en este juego, los productores de una famosa cadena de televisión han decidido reinventarlo, llamando *niveles* a lo que antes eran paredes, publicitando el programa a través de *influencers* y cambiando el nombre del concurso a ‘*El juego de las puertas*’. Pero los cambios más importantes se han hecho con el propósito de cumplir con las medidas de protección y seguridad actualmente vigentes para participantes en concursos televisivos.

La *Comisión de Protección al Participante (CPP)* considera que la versión original del juego tiene dos grandes inconvenientes. Por un lado, es propenso a causar lesiones (por razones obvias), mientras que por otro lado, tener cuatro opciones entre las que elegir es propenso a aumentar el nivel de estrés de los participantes. Es por esto que no solo se ha reducido el número de puertas por nivel a dos, sino que ahora ninguna puerta es falsa; todas están hechas de madera y se debe pasar por ellas andando, para evitar accidentes.

Para que el juego siga teniendo algo de emoción, se ha introducido una nueva mecánica: las llaves de colores. Cuando comienza el juego, cada participante tiene una (y solo una) llave de cada color. Por otro lado, cada puerta tiene un número variable de cerraduras de distinto color (al menos una y no más de 10), y para poder abrir la puerta y pasar al siguiente nivel, el participante debe desbloquear todas las cerraduras de la puerta utilizando llaves de su mismo color. Puede haber cerraduras del mismo color en distintas puertas. Eso sí, al igual que en los videojuegos, las llaves se consumen cuando se utilizan, lo que hace que, por ejemplo, un participante pueda desbloquear una única cerradura roja en todo el juego, lo que requiere tener algún tipo de estrategia si quiere evitar la situación en la que no pueda avanzar más por no tener las llaves necesarias.

Sabiendo que el concurso es individual y que, por tanto, los participantes no pueden intercambiarse llaves, ¿puedes diseñar una estrategia que permita a un único participante avanzar lo máximo posible?

Entrada

La entrada comienza con el número de casos de prueba que vendrán a continuación, cada uno describiendo una posible configuración del juego de las puertas.

La primera línea de cada caso de prueba indicará el número N ($1 \leq N \leq 10^4$) de niveles que tiene el juego.

Los siguientes N pares de líneas del caso de prueba describirán cada uno de los niveles del juego, en el mismo orden en el que deben superarse: la primera línea de cada par contendrá la descripción de la puerta izquierda y la segunda línea la descripción de la puerta derecha.

La descripción de una puerta consiste en el número M_i ($1 \leq M_i \leq 10$) de cerraduras que tiene, seguido de M_i valores distintos X_{ij} ($0 \leq X_{ij} \leq 10^6$). Cada número X_{ij} identifica de forma única a un color.

Se garantiza que en un mismo juego no habrá más de 10 cerraduras del mismo color.

Salida

Por cada caso de prueba se describirá en una única línea una estrategia que nos permita llegar lo más lejos posible como una serie de caracteres I / D, indicando para cada nivel, qué puerta debería abrirse (Izquierda o Derecha).

Si hay varias estrategias óptimas, se imprimirá cualquiera de ellas.

Entrada de ejemplo

```
2
3
2 5 6
2 3 6
3 1 2 7
1 6
2 3 7
2 4 5
2
2 1 2
2 1 2
1 1
1 2
```

Salida de ejemplo

```
DID
D
```

B

¡Hoy toca pisci!

Mi sobrina Marta está encantada porque este trimestre en el cole los de tercero han empezado a ir a la piscina para aprender a nadar. Algunos saben algo más y otros algo menos, pero todos se divierten por igual. Marta espera ansiosa que llegue el jueves, porque es el día que van a la piscina. Además de lo que supone chapotear en el agua, a la piscina van en autobús desde el cole, y después de estar en la piscina toman un bocadillo al sol, por lo que es toda una excursión.



El problema es que en tercero son tres clases, y por el aforo de la piscina, solamente pueden ir dos clases cada vez. Así que en el colegio han decidido que cada jueves van dos clases distintas y una descansa. Un jueves descansa el grupo A, al siguiente el grupo B y al tercero descansa el grupo C. Y luego vuelven a empezar.

El conductor del autobús sabe cuántos alumnos ha llevado cada uno de los tres últimos jueves y se pregunta si con eso podría calcular el tamaño de cada uno de los tres grupos. Los jueves de piscina ningún alumno falta al cole. ¿Puedes ayudarle?

Entrada

La entrada comienza con el número de casos de prueba que vendrán a continuación.

Cada caso aparece en una línea y consta de tres números, la cantidad de alumnos que fueron en el autobús cada uno de los tres últimos jueves. En el autobús no caben más de 50 alumnos.

Salida

Por cada caso de prueba se escribirá una línea con los tamaños de los tres grupos de tercero del cole de Marta, ordenados de menor a mayor.

Entrada de ejemplo

```
3
40 40 40
44 49 45
52 32 42
```

Salida de ejemplo

```
20 20 20
20 24 25
11 21 31
```


C

Juez amable

En la mayoría de los concursos de programación y jueces en línea para entrenar, cuando un participante/usuario hace un envío el veredicto del juez suele ser bastante poco amable. Si la solución enviada es correcta se consigue un “*ACCEPTED*” (AC). Si la solución no es capaz de responder correctamente a todos los casos de prueba, entonces se tiene un veredicto genérico como “*WRONG ANSWER*” (WA) o “*TIME LIMIT*” (TLE). Pero esa información es bastante vaga porque no dice nada sobre cómo de cerca está la solución de ser correcta. Al fin y al cabo no es lo mismo haber fallado en todos los casos de prueba que solo en unos pocos.

Dar la información de cuántos casos son incorrectos puede no ser fácil. Cuando en cada ejecución de la solución se procesa un único caso entonces es muy fácil: basta con contar cuántas ejecuciones no responden bien. En el otro extremo, cuando una única ejecución procesa todos los casos de prueba, la ejecución fallida podría estar causada únicamente por uno de ellos o por todos.

La *Asociación de Estudiantes Rabiosos* lleva años pidiendo que mejore la tecnología de los sistemas de concursos para dar información adicional sobre los casos incorrectos. Tras años de escuchar las reivindicaciones, los técnicos de la *Unión de Competiciones y Maratones* han decidido coger el toro por los cuernos. El primer paso ha sido programar un juez que en lugar de simplemente aceptar las soluciones a los problemas, permite además indicar qué intervalo de casos de prueba se quieren probar. Entonces el juez dice si la solución responde correctamente a todos los casos del intervalo (AC) o no (WA).

Con esto lo único que queda por hacer es una aplicación que lance al juez las consultas necesarias para poder determinar cómo de cerca está la solución de ser correcta.

La aplicación se ejecutará para un número indeterminado de soluciones de participantes. Para cada una leerá por la entrada estándar el número n de casos de prueba que tiene el problema en cuestión (entre 1 y 10^6). Entonces la aplicación escribirá una línea por cada una de las ejecuciones que quiera hacer de la solución. La línea comenzará por el signo de interrogación “?” al que seguirán dos números a y b ($1 \leq a \leq b \leq n$). El juez responderá con una línea con el texto “AC” si la solución contestó correctamente a todos los casos entre a y b (ambos incluidos) o “WA” en caso contrario.

El objetivo es ser algo más amables que los jueces habituales pero no muchísimo más. Si la aplicación determina que la solución es completamente correcta, escribirá una línea “Veredicto: AC”. Si determina que el número de casos de prueba con los que la solución falla es cuatro o más, escribirá “Veredicto: WA”. Por último, si el número de casos incorrectos está entre 1 y 3, listará sus números en orden creciente en una línea con la forma: “Veredicto: WA en [numCaso1] [numCaso2] [numCaso3]” (entre cada una de las palabras hay siempre un único espacio).

Nuestra aplicación terminará cuando llegue a un envío cuyo problema no tenga casos de prueba.

A continuación aparece un ejemplo de ejecución. En cursiva aparece lo escrito por la máquina y en negrita lo escrito por una posible aplicación.

```
100
? 1 100
AC
Veredicto: AC
100
? 1 10
WA
? 11 20
WA
? 21 30
WA
? 31 40
WA
Veredicto: WA
```

CPP	TIMELIMIT
CPP	CORRECT
CPP	CORRECT
CPP	WRONG-ANSWER ?
CPP	TIMELIMIT

```
100
? 2 2
WA
? 1 1
WA
? 3 100
AC
Veredicto: WA en 1 2
0
```

Notas

Al tratarse de un problema *interactivo* es importante que cada vez que tu solución escriba algo que el juez deba leer se asegure de volcar la salida (usando terminología inglesa, haga “*flush*”). La forma de hacerlo varía entre lenguajes. En los admitidos en la competición puede hacerse con:

- C++: `cout << endl;` o `cout << flush;`
- C: `fflush(stdout);`
- Java: `System.out.flush();`
- Python: `print(..., flush=True)` o `sys.stdout.flush()`

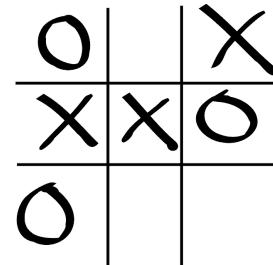
● D

Tres en raya

A los hermanos Rayo les encanta el juego de las tres en raya, tanto que no pueden parar de jugarlo.

Para jugar al juego de las tres en raya hacen falta dos personas y un tablero de dimensiones 3×3 . Un jugador utiliza el símbolo X mientras que el otro utiliza el símbolo O, y se van turnando para colocar su símbolo en una casilla del tablero que no contenga ya uno de los símbolos. El juego termina cuando no es posible colocar más símbolos, o cuando uno de los jugadores gana al conseguir que en el tablero se encuentren 3 de sus símbolos dispuestos en una línea recta. En cuanto el juego termina, se quitan todos los símbolos del tablero para que pueda dar comienzo una nueva partida.

Cuando se cansan de jugar, los hermanos Rayo ponen pausa a la partida actual con la intención de continuarla más adelante. El problema es que cuando vuelven a jugar más tarde, no suelen acordarse de cómo iba la partida. Lo único que saben es quién la había empezado. ¿Puedes ayudarles?



Entrada

La entrada comienza con el número de casos de prueba que vendrán a continuación. Cada caso muestra el estado del tablero que se encuentran los hermanos Rayo cuando deciden continuar una partida que dejaron a medias.

La primera línea de cada caso de prueba contiene un único símbolo ‘X’ o ‘O’, indicando cuál de los dos símbolos tuvo el primer turno en esa partida.

A continuación aparece el estado del tablero a lo largo de 3 líneas con 3 caracteres cada una, separados por espacios. Los caracteres podrán ser ‘X’, ‘O’ o ‘.’. Los dos primeros indican que la casilla contiene ese símbolo y el punto indica que la casilla no contiene ningún símbolo.

Salida

Por cada caso de prueba, el programa indicará en una línea independiente cómo deben continuar la partida los hermanos Rayo, indicando de quién es el siguiente turno de la siguiente manera: SIGUE X o SIGUE O.

Si el juego justo había terminado cuando lo dejaron, se indicará el ganador de la siguiente manera: GANA X o GANA O. Y si el juego había terminado en un empate se escribirá: EMPATE.

Cuando sea imposible haber alcanzado ese estado del tablero sin saltarse alguna norma, se escribirá IMPOSIBLE.

Entrada de ejemplo

```
3
X
X . X
. X O
O . .
O
X . O
X O .
X . O
X
. . .
. . .
X O O
```

Salida de ejemplo

SIGUE O
GANAN X
IMPOSIBLE

● E Pintoresca vieja Europa

Cuatro años llevaba esperando a que llegara este momento. Más concretamente desde que su hermana mayor terminó secundaria en un pueblecito cercano a la capital de Argentina. La posición acomodada de su familia y la importancia que sus padres concedían a conocer mundo hicieron que le regalaran a su hermana, como premio por sus buenas notas, un viaje por Europa al que se unió un grupo de amigos. Partió en un vuelo transatlántico con una mochila llena de sueños y tras una ruta circular por Europa volvió con los bolsillos llenos de experiencias inolvidables.



Su padre le había prometido que él podría hacer un viaje similar y llevaba cuatro años soñando con que llegara el momento y planificando esa ruta circular viajando en tren de ciudad en ciudad en compañía de sus mejores amigos. Pero la cosa se ha ido al traste, porque sus notas no han sido las de su hermana. Sus padres quieren que se vaya, claro, (“*es importante conocer mundo, cariño*”, le dicen) pero no están dispuestos a sufragar todos los gastos (“*Che, tu hermana en la mochila llevaba sueños, vos además cargás con tus suspensos, loco.*”). Han aceptado pagarle el viaje de avión de ida y vuelta a la ciudad europea que elija pero el resto de billetes de tren los tendrá que pagar él con los trabajillos que pueda ir haciendo por el camino.

Entrada

La entrada estará formada por distintos casos de prueba, cada uno ocupando tres líneas.

La primera línea contiene el número n de ciudades de la ruta circular (hasta 300.000). La siguiente línea tiene n números con el dinero que podrá conseguir con trabajos puntuales en cada una de ellas, tras descontar lo que se gasta en comida y alojamiento. En la tercera línea aparece el coste del viaje para ir a la siguiente ciudad. Ten en cuenta que el primer valor será el precio del billete para ir de la ciudad 1 a la ciudad 2 y que el último número será el coste del billete desde la ciudad n a la ciudad 1. Todas las cantidades están entre 0 y 100.000.

La entrada termina con una línea con un cero que no debe procesarse.

Salida

Por cada caso de prueba se escribirá una línea con la lista de ciudades desde las que se podría iniciar la ruta circular terminándola sin caer en números rojos en ningún punto de la ruta. Las ciudades se darán separadas por comas y ordenadas por la posición que ocupan en la entrada y se condensarán en bloques las ciudades con identificadores consecutivos, separando el identificador más bajo y el más alto por un guión.

Si ninguna ciudad puede utilizarse como punto de partida, se escribirá NO HAY VIAJE.

Entrada de ejemplo

```
6
8 0 8 0 8 0
4 4 4 4 4 4
4
10 10 0 4
1 4 6 3
4
5 5 5 5
7 7 7 7
0
```

Salida de ejemplo

```
1, 3, 5  
1-2, 4  
NO HAY VIAJE
```

F

Preparándose para hibernar

Se acaba el otoño en Yellowstone y Yogui, como todos los años, lo deja todo para el último día. Pronto bajarán considerablemente las temperaturas, por lo que se acerca el momento idóneo para echar una cabezadita hasta la primavera. El problema es que uno no se puede echar una siesta de seis meses con el estómago vacío y nuestro amigo no ha comido nada desde el banquete de miel que se dio hace dos días.

Con la ayuda del guardabosques Smith, que le ha facilitado un mapa del bosque, Yogui conoce todas las áreas de picnic cercanas, los caminos que las unen entre sí (y con su *osera*), y la cantidad de comida que “se puede quedar olvidada” en cestas en algunas de estas áreas.

Yogui quiere conseguir la máxima cantidad de comida y para eso cuenta con la ayuda de su amigo Bubu. En cada misión de recuperación, que siempre comienza desde su osera común, cada oso camina hasta el área que haya elegido donde recoge las cestas “olvidadas” y vuelve a la osera a dejar su botín antes de emprender el camino hacia otra área (no puede empezar a “recuperar” cestas en un área llevando ya cestas de otra).

¿Cuánto podrán comer como máximo tras este único día de recolección, sabiendo que cuando se haga de noche los dos osos quieren estar de vuelta en la osera para poder cenar?



Entrada

La entrada está compuesta por diversos casos de prueba. En la primera línea aparece ese número de casos.

Cada uno de ellos comienza con una línea con cuatro números enteros: A (entre 1 y 100) que representa el número de áreas de picnic que hay en Yellowstone; C (entre 0 y 5.000) que representa el número de caminos que hay en el bosque; T (entre 0 y 500) que representa el número de unidades de tiempo que quedan para que se haga de noche; y N (entre 0 y A), que representa el número de áreas de picnic donde saben que hay comida “olvidada”.

En la siguiente línea aparecen N pares de números. De cada par, el primer número (entre 1 y A) representa un área de descanso (todas distintas) y el segundo la cantidad de comida que se puede recuperar ahí. Sabemos que la suma total de comida no es mayor que 10^9 .

Después aparecen C líneas que representan los caminos con tres enteros: los dos primeros son los puntos que une y el tercero las unidades de tiempo que se tarda en recorrer ese camino (un valor entre 1 y 100). La osera se representa con el número 0, y las áreas de picnic se numeran de 1 a A . Todos los caminos se pueden recorrer en ambas direcciones.

Salida

Para cada caso de prueba se escribirá una línea con un entero que indique la máxima cantidad de comida que los osos podrán recoger antes de que se haga de noche.

Entrada de ejemplo

```
1
5 6 45 3
2 20 3 10 5 100
0 1 5
1 2 20
0 3 5
2 4 5
3 4 10
4 5 5
```

Salida de ejemplo

120

● G Willy

Willy es un zángano que acostumbra acompañar a la abeja Maya en sus excursiones para recolectar polen. Como es muy goloso busca siempre las flores que tienen más granos de polen. Si suponemos que las flores del campo donde van Maya y Willy se distribuyen sobre una cuadrícula, Willy se mueve siempre hacia la flor adyacente en una de las cuatro direcciones de la cuadrícula que tiene más polen. Si hay dos flores que tienen la misma cantidad de polen elige aquella que se encuentra antes según se mueven las agujas del reloj desde las 12:00. Cuando ninguna de las flores adyacentes tiene polen, Willy se tumba a dormir hasta que Maya termina con su recolección.

Sabiendo cómo recolecta el polen su amiguito, Maya quiere saber rápidamente la flor en la que debe empezar Willy a recolectar para obtener la mayor cantidad de polen posible.



Entrada

La entrada comienza con el número de casos de prueba que se dan a continuación. Cada caso comienza con una línea que indica el tamaño de la cuadrícula donde recogen el polen (número de filas y número de columnas). A continuación se muestran los granos de polen que hay en cada posición de la cuadrícula.

El tamaño máximo de la cuadrícula es de 30 por 30 posiciones y nunca hay más de 100 granos de polen en cada flor. Willy puede empezar a recolectar el polen en cualquier posición de la cuadrícula.

Salida

Para cada caso de prueba se escribirá en una línea el número máximo de granos de polen que puede recolectar Willy.

Entrada de ejemplo

```
3
1 1
5
3 4
2 1 1 9
0 1 0 0
0 9 0 0
4 5
0 0 1 0 0
9 1 1 1 9
0 0 1 0 0
0 0 9 0 0
```

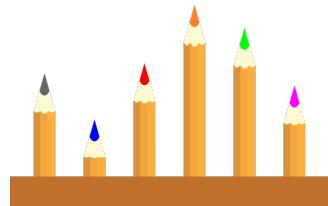
Salida de ejemplo

```
5
13
12
```


H

Lápices de colores

Bosco se ha comprado un berbiquí y un trozo de madera en una ferretería para fabricarse un portalápices con el que tener organizados sus lápices de colores. Con ayuda de la regla, ha llenado la longitud del madero con puntos, cada uno de ellos separado un centímetro del anterior, que luego ha perforado hábilmente con su nueva herramienta. Una vez terminado, ha colocado el invento sobre su escritorio y ha encajado tantos lápices de colores como ha podido. En su diseño original la idea era poner la punta hacia abajo en los pequeños agujeros hechos con el berbiquí pero pronto se ha dado cuenta de que le han quedado unos orificios demasiado grandes y tiene que meterlos con la punta hacia arriba para que se sujeten.



Lejos de entristercerse por el fallo, ha resuelto darle una nueva utilidad a su creación como trazador de líneas paralelas. El problema es que algunos lápices están más gastados que otros y no tienen la misma longitud, con lo que no hay manera de pintar con todos a la vez. Una vez colocados los lápices en el portalápices, ¿cuál es el máximo número de líneas paralelas que puede pintar con ellos? Se supone que puede girar el instrumento como quiera para apoyarlo sobre el papel.

Entrada

La entrada comienza con el número de casos de prueba que vendrán a continuación.

Cada caso ocupa dos líneas: la primera indica el número de agujeros que contiene el portalápices (separados entre sí por un centímetro) y la segunda linea enumera las longitudes de esos lápices en centímetros, de izquierda a derecha. En el portalápices no caben más de 5.000 lápices y las longitudes de estos varían entre 1 y 10.000.

Salida

Por cada caso de prueba se escribirá una línea con el máximo número de líneas que se pueden pintar en paralelo.

Entrada de ejemplo

```
2
3
2 1 2
5
1 2 3 4 2
```

Salida de ejemplo

```
2
4
```


● | Miguelín Duráin

Miguelín Duráin, ciclista retirado, se ha apuntado este año al *tour de los veteranos*. Es una carrera muy irregular, porque hay algunos corredores que acaban de terminar su vida profesional y todavía están en muy buena forma, y hay abuelitos legendarios que se bajan de la bici para subirse al andador.

Una de las etapas será una contrarreloj. En ella, los corredores van tomando la salida de uno en uno, y deben enfrentarse a la carrera en solitario. Cuando un corredor va suficientemente rápido puede terminar adelantando a corredores que salieron antes que él. En el *tour de los veteranos* esto ocurre con cierta frecuencia por la diferencia de forma de los participantes.



Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea en la que se indica el número de ciclistas que tomarán la salida y los minutos que pasarán entre cada salida. En la línea siguiente se dan los tiempos que tarda cada ciclista en completar la etapa en el orden en que tomarán la salida. La entrada termina con dos ceros que no deben procesarse.

El número de ciclistas que participan en la contrarreloj es positivo y menor que 300.000. El tiempo que tardan los ciclistas en completar la carrera es mayor que cero y menor que 10^6 y los minutos de diferencia entre cada ciclista es un valor mayor que cero y menor que 1.000.

Salida

Para cada caso de prueba se escribirá en una línea el número de adelantamientos que se producen en la carrera. Ten en cuenta que no se considera adelantamiento cuando dos ciclistas llegan simultáneamente al final de la etapa.

Entrada de ejemplo

```
6 1
4 6 3 5 3 4
3 2
6 4 2
4 2
10 7 4 1
0 0
```

Salida de ejemplo

```
2
0
6
```


J

Jornada partida

Cuando a Aitor le ofrecieron un trabajo a jornada partida como conserje en el polideportivo, se hizo ilusiones pensando que tendría tiempo libre a mediodía para irse a su casa a comer y luego echarse una siesta. Pero la realidad fue bien distinta. Lo que sus jefes llamaban *jornada partida* consistía en que Aitor podía irse a su casa cuando no hubiera ningún usuario en el polideportivo, pero si había alguien allí, debía quedarse. Para colmo, el polideportivo estaba disponible las 24 horas del día, 7 días a la semana! Más de una vez, Aitor ha tenido que levantarse en mitad de la noche para ir al polideportivo por culpa de algún deportista que entrenaba a las tres de la madrugada.



Pese a estos casos extremos, el polideportivo se quedaba vacío a menudo, por lo que Aitor siempre aprovechaba la oportunidad de irse a su casa, aunque fuera solamente para acariciar a su perro y volverse inmediatamente después. No obstante, si preveía que no le iba a dar tiempo a ir y volver antes de que llegara el siguiente usuario, prefería quedarse en el polideportivo y esperarlo allí. Al fin y al cabo, no tenía mucho sentido irse a casa para tener que volverse a mitad de camino.

A Aitor le ofrecen renovar el contrato para la siguiente temporada, pero no sabe si le compensa tanto viaje de ida y vuelta a su casa. Tiene un listado detallado de las personas que visitarán el polideportivo durante la temporada, y una previsión sobre cuánto tiempo estarán entrenando allí. ¿Le ayudas?

Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba comienza con el número N de visitas de usuarios que tendrá el polideportivo a lo largo de la temporada y el número M de minutos que tarda Aitor en desplazarse desde el polideportivo hasta su casa ($1 \leq N \leq 500.000$, $1 \leq M \leq 200$). Después vienen N líneas, una por cada visita. Cada línea indica la fecha/hora de entrada, y la fecha/hora de salida de la visita. Cada fecha se expresa mediante un número D ($0 \leq D < 365$) que indica el número de días transcurridos desde el comienzo de la temporada. Cada hora tiene el formato HH:MM, donde HH indica la hora (entre 00 y 23) y MM indica el minuto (entre 00 y 59).

Suponemos que, para cada visita, el usuario entra y sale justo en el momento en el que el reloj marca la respectiva hora de entrada/salida (es decir, en el segundo 0), que la fecha y hora de salida siempre es posterior a la de entrada, y que Aitor tarda la misma cantidad de minutos desde su casa hasta el polideportivo que desde el polideportivo hasta su casa. Además, suponemos que él comienza la temporada en su casa y que la primera visita es lo suficientemente tarde como para que le dé tiempo a llegar al polideportivo para atenderla.

La entrada finaliza con dos ceros (0 0), que no se procesan.

Salida

Para cada caso de prueba debe escribirse una línea con el número de viajes que Aitor tendrá que hacer desde su casa hasta el polideportivo durante toda la temporada.

Entrada de ejemplo

```
4 10
2 15:00 2 15:45
2 15:30 2 16:50
2 17:10 2 18:10
3 10:00 3 10:10
3 20
1 03:39 1 03:40
0 23:00 1 02:00
1 02:00 1 03:00
0 0
```

Salida de ejemplo

3
1

K

Campamento de verano

Susana es monitora en el campamento de verano *FAQ*, de Física, Algoritmia y Química. Cada verano llegan al campamento, desde distintos puntos del país, cientos de estudiantes ávidos de pasárselo bien durante unas semanas y adquirir nuevos conocimientos avanzados sobre sus temas favoritos.

Todos los días realizan varias competiciones donde se enfrentan a diferentes retos y resuelven complicados problemas con los que van consiguiendo puntos que al final del campamento se pueden transformar en valiosos premios.

El primer día los participantes no se conocen entre sí y participan de manera individual. Sin embargo, según se van conociendo tienen permitido (para promover el compañerismo y el trabajo en equipo) unirse formando equipos. Cuando un equipo resuelve un problema todos los miembros del equipo reciben la misma cantidad de puntos. Eso sí, una vez que un equipo está constituido no puede dividirse, solamente puede seguir igual o unirse a otro equipo para formar uno nuevo aún mayor. Respecto a los puntos conseguidos hasta la unión no hay cambios, cada integrante conserva los puntos que tenía.

Susana es la encargada de gestionar cuántos puntos ha conseguido cada participante hasta el momento. Y sabe que no es nada fácil, ya que el campamento tiene mucho éxito y cada vez es mayor el número de participantes, por lo que está pensando que es hora de hacer un programa que la ayude con la gestión. ¿La puedes ayudar?



Entrada

La entrada comienza con el número de casos de prueba que vendrán a continuación. Cada caso comienza con una línea con dos números enteros: N , el número de participantes en el campamento (numerados de 1 a N), y M , el número de acciones que Susana tiene que llevar a cabo respecto a la gestión de los puntos ($1 \leq N, M \leq 200.000$).

Las siguientes M líneas contienen las acciones, que pueden ser de tres tipos: U seguido de dos identificadores de estudiantes A y B , con $1 \leq A, B \leq N$, que indica que los equipos (posiblemente unitarios) a los que pertenecen los estudiantes A y B se han unido y a partir de ese momento participarán en las pruebas siempre juntos; S seguido de un estudiante A y un valor V , con $1 \leq A \leq N$ y $1 \leq V \leq 500$, que indica que todos los integrantes del equipo al que pertenece el estudiante A han recibido V puntos que se suman a los puntos que llevaba cada uno; y P seguido de un estudiante A , con $1 \leq A \leq N$, que indica que se quiere saber cuántos puntos tiene en ese momento el estudiante A .

Salida

Para cada caso de prueba se escribirá una línea por cada acción P con los puntos en ese momento del estudiante consultado, teniendo en cuenta que inicialmente todos los estudiantes comienzan con 0 puntos y forma cada uno de ellos un equipo unitario.

Tras el tratamiento de cada caso de prueba se escribirá una línea más con tres guiones, ---.

Entrada de ejemplo

```
1
3 7
S 1 10
S 2 20
U 1 2
S 2 25
P 1
P 2
P 3
```

Salida de ejemplo

```
35  
45  
0  
---
```

● L ¡Vamos a la feria!

Este fin de semana han sido las fiestas de Una Ciudad Madrileña y ha habido multitud de actividades para que sus habitantes pudieran disfrutarlas. Como suele ser habitual, el ayuntamiento aprovechó una gran explanada para situar la feria: había cantidad de puestos de comida y bebida, rifas... y cómo no, atracciones!

En una de las atracciones más antiguas y famosas de la feria se entra siempre por grupos. Cada grupo puede estar formado por niños y adultos. El precio de la entrada infantil puede ser igual o distinto del precio de la entrada de adulto pero, por simplicidad, han decidido que ambos precios sean siempre una cantidad de euros que sea un número entero mayor o igual que cero.



Y no solo la atracción lleva muchos años en la feria, sino que los encargados de la atracción llevan ya muchísimos años regentándola. Y sus costumbres no se han modernizado nada... incluso siguen tomando nota de los grupos que entran a mano, anotándolas sencillamente en unos cuadernos.

Hemos encontrado uno de esos cuadernos tirado en el suelo cerca de la atracción, y mientras íbamos a devolvérselo a los feriantes, lo hemos hojeado curioseando los precios. En él estaban escritos los grupos que han entrado en la atracción, especificando cuántos adultos entraron en cada grupo, cuántos niños y el precio total de las entradas asociadas a ese grupo. Con dicha información, ¿podríamos haber determinado cuánto cuesta cada entrada (distinguiendo las de adultos y las de niños)?

Entrada

La primera línea de la entrada es un entero que indica cuántos casos de prueba se han de tratar. Cada caso de prueba representa un cuaderno que nos hemos encontrado.

En cada caso de prueba, la primera línea tiene un entero n ($1 \leq n \leq 100.000$), que indica el número de grupos que hay apuntados en el cuaderno. Le siguen n líneas, cada una representando a un grupo. Cada una de esas líneas contiene tres enteros (entre 0 y $2 \cdot 10^9$) con la cantidad de adultos que entraron en ese grupo, la cantidad de niños de ese grupo y el precio total de todas las entradas de ese grupo. En cada grupo hay, al menos, una persona (ya sea adulto o niño).

Salida

Por cada caso de prueba, si es posible determinar el precio de las entradas de adulto y niño, se sacarán por pantalla dos enteros, indicando dichos precios. Si no hubiera suficiente información como para poder deducir dichos precios, se imprimirá una línea con el texto NO HAY SUFICIENTES DATOS. En cambio, si puedes deducir que los feriantes cometieron algún error al anotar los grupos de ese cuaderno, se escribirá una línea con el texto CUADERNO CON ERRORES.

Entrada de ejemplo

```
4
3
2 1 11
3 0 9
1 5 28
2
1 1 1
0 1 1
2
0 2 6
2 1 8
1
2 1 11
```

Salida de ejemplo

```
3 5  
0 1  
CUADERNO CON ERRORES  
NO HAY SUFICIENTES DATOS
```