"First, solve the problem. Then, write the code" John Johnson.

ProgramaMe 2023
Final Nacional
Problemas

Patrocinado por

Ejercicios realizados por

Universidad Complutense
de Madrid

Realizado en la **Facultad de Informática (UCM)**
9 de junio de 2023

9 de junio de 2023
https://www.programame.com

Listado de problemas

Autores de los problemas:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)
- Manuel Montenegro Montes (Universidad Complutense de Madrid)
- Antonio Pérez-Aradros Herrero (IES Comercio)
- Javier del Río López (Universidad Complutense de Madrid)

Revisores:

- Isabel Pita Andreu (Universidad Complutense de Madrid)
- Alberto Verdejo (Universidad Complutense de Madrid)

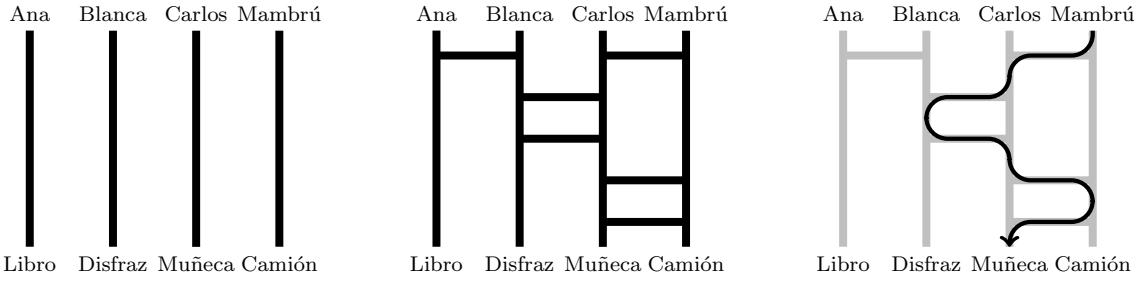
A
Cumpleaños en el jardín de infancia

Las dinámicas que se crean en las aulas de las escuelas infantiles alrededor de los cumpleaños de los críos son una locura. Para que no haya envidias ni se creen conflictos entre ellos, todos los niños celebran su cumpleaños e invitan a todos los demás de la clase. Eso supone un caos de regalos que crece de forma exponencial con su tamaño.

Para poner fin a esta locura, Justo Reparto, el padre de una de las niñas que asiste al jardín de infancia "*Acepto el retoño*" ha tenido una idea para el curso que viene. En lugar de tanta celebración, ha propuesto hacer solo una, aunque sea a lo grande, en el que todo el mundo se dé por celebrado, felicitado y regalado. Para eso, cada niño aportará a la celebración un regalo. Luego los regalos se repartirán entre todos, de modo que a cada peque le toque uno, sea el que sea. De esa forma también aprenderán que no hay juguetes de niño y juguetes de niña, porque podría ocurrir que su hija terminara llevándose un camión de bomberos, y el pequeño Mambrú Tote una muñeca.

Aceptada la idea, se ha decidido que para repartir los regalos entre los niños, se usará un *amidakuji* (o *Ghost leg*), un juego japonés de reparto de n objetos entre n personas. Se comienza con una hoja de papel en la que se escriben en la parte superior los nombres de las personas, y, alineados en la parte inferior, los objetos, conectando con una línea vertical un objeto con una persona. A continuación se incorporan, de forma arbitraria, líneas horizontales para conectar dos líneas verticales adyacentes, garantizando, eso sí, que no haya líneas horizontales que se toquen.

Para saber qué regalo le toca a cada uno, se *siguen las líneas* de arriba hacia abajo sabiendo que cada vez que se alcanza una línea horizontal *hay que seguirla* para cambiar de columna de descenso.



Paso 1: líneas verticales | Paso 2: líneas horizontales | Paso 3: resolución

Para evitar que se haga trampa, Justo Reparto pondrá las líneas verticales, los nombres y los regalos en el orden que le parezca, y los tapará. Luego los niños pondrán las líneas horizontales sin saber qué están uniendo. Justo tiene la idea, aunque no lo haya dicho, de colococar en la misma columna a cada niño y el regalo que ha traído, de modo que, si no se ponen líneas horizontales, todos se vayan a casa con su propio regalo. Además, colocará los regalos ordenados por su valor empezando, a la izquierda, con el más barato.

Entrada

La entrada comienza con un número que indica cuántos casos de prueba tendrán que procesarse. Cada uno comienza con un número $2 \leq N \leq 200.000$ indicando cuántos niños asisten al cumpleaños, cada uno con un regalo.

Después aparecen N–1 líneas, una por cada columna del amidakuji, de izquierda a derecha, salvo para la última. Cada una comienza con el número de líneas horizontales que salen de esa columna hacia la siguiente a su derecha. A continuación aparece un número por cada conexión, indicando la distancia a la que está de la parte superior de la figura.

En ningún caso hay más de 200.000 líneas horizontales, y ninguna está a una distancia mayor de 10^9 del inicio de las columnas. Todas las conexiones son completamente horizontales y se garantiza que

no existirá ninguna columna con dos o más conexiones a la misma altura (hacia la izquierda o hacia la derecha).

Salida

Por cada caso de prueba se escribirá una línea con tres números separados por espacio. El primero indica cuántos niños obtienen un regalo *peor* que el que aportaron (queda *a la izquierda* de su posición), el segundo indica cuántos se llevan el regalo que llevaron a la celebración, y el tercero cuántos se llevan un regalo *mejor* que el que entregaron (queda *a la derecha*).

Entrada de ejemplo

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

Salida de ejemplo

```
2 0 2
1 0 1
2 1 1
```

B
Efecto rebote

He probado de todo. La dieta de la alcachofa, de la manzana, del repollo, del semáforo, de la luna... Pero nada. Con todas he sufrido el *efecto rebote*. Después de realizar un esfuerzo increíble para adelgazar, lo que incluye hasta pasar hambre, cuando alcanzo el peso objetivo y recupero mi dieta normal, el cuerpo reacciona recuperando, ¡y a qué velocidad!, el peso perdido. Y los males no terminan ahí. A veces incluso, coge carrerilla, y supera con creces el peso original.

Y ya digo que hablo por experiencia. Llevo mucho tiempo anotando religiosamente mi peso todas las mañanas, y el resultado es una gráfica que parece la de un sismógrafo en la falla de San Andrés.

Entrada

El programa leerá, de la entrada estándar, múltiples casos de prueba. Cada uno estará compuesto por un primer número indicando el número de días, $2 \leq d \leq 1.000.000$, en los que se ha registrado el peso. A continuación aparecerán d números (menores que 2.000.000) indicando el peso, en gramos, de cada uno de esos días. Se garantiza que el peso en dos días consecutivos nunca será el mismo.

La entrada termina con un caso de prueba sin pesos, que no deberá procesarse.

Salida

Para cada caso de prueba el programa escribirá la longitud, en días, del periodo más largo de régimen seguido de su efecto rebote, es decir un descenso continuado del peso, seguido de un ascenso ininterrumpido.

Entrada de ejemplo

```
5
10 8 5 9 12
8
7 8 10 14 24 19 20 21
3
10 7 5
0
```

Salida de ejemplo

```
5
4
0
```

C
El bingo de Azahara

El bingo es un juego de azar en el que cada jugador dispone de una o varias tarjetas, llamadas *cartones*. Cada cartón tiene impresa una serie aleatoria de números. Por otro lado, un locutor extrae sucesivamente varias bolas numeradas de un bombo, anunciando los números extraidos. Cada vez que el locutor anuncia un número, los jugadores lo tachan de sus cartones, en caso de tenerlo. El primer jugador que consigue tachar todos los números de alguno de sus cartones grita "*¡Bingo!*" y gana el juego. Si varios jugadores completan el cartón al mismo tiempo, todos ganan la partida.

Azahara Fortunata ha decidido montar un salón con una variante de este juego, llamada *Bingo Injusto*. En esta variante, cada jugador dispone de un único cartón, pero la cantidad de números que contiene el cartón es aleatoria. De este modo, puede haber jugadores que tengan más números que tachar que otros. Como Azahara no dispone de mucho presupuesto, ha decidido ella misma tomar el papel de locutora del juego. El problema es que le aburre tener que comprobar cuáles de los cartones son los premiados, y ha decidido hacer un programa que realice esta ingrata tarea por ella. ¿Puedes ayudarla?

Entrada

La entrada consta de una serie de casos de prueba, cada uno de ellos describiendo una partida de bingo. Cada caso de prueba comienza con el número N de jugadores participantes ($1 \leq N \leq 50.000$). A continuación aparecen N líneas, cada una describiendo el cartón de cada jugador. Cada una de estas líneas comienza con el nombre del jugador (secuencia de como mucho 40 letras minúsculas del alfabeto inglés) y va seguida por la secuencia de números contenidos en el cartón de dicho jugador (números entre 1 y 10^6, todos distintos). Esta secuencia finaliza con el número 0, que no forma parte del cartón. Ningún cartón tiene más de 1.000 números.

A continuación aparece la descripción de la secuencia de números que salen del bombo, todos distintos, que son los que Azahara irá leyendo hasta que algún/algunos de los jugadores grite *¡Bingo!*. En la primera línea aparece la cantidad de números y en la siguiente aparecen esos números, separados por espacios. Suponemos que no hay ningún jugador despistado que se olvide de tachar un número de su cartón, o que se olvide gritar *¡Bingo!* cuando consigue tachar todos los números de su cartón. Pero en la secuencia sí puede haber más números de los necesarios para encontrar un ganador, que salieron del bombo y estaban preparados por si hicieran falta.

La entrada finaliza con una partida de 0 jugadores, que no se procesa.

Salida

Para cada caso de prueba se escribirá una línea con los nombres de los jugadores que han ganado la partida. La lista de ganadores ha de estar ordenada alfabéticamente de manera ascendente. Si una partida tiene varios ganadores, sus nombres deben estar separados por un espacio.

Entrada de ejemplo

```
2
diana 30 21 23 32 0
martin 20 21 31 46 78 23 0
6
20 23 21 32 46 30
3
felipe 15 9 21 0
ricardo 10 2 9 0
gerardo 7 9 0
7
15 21 10 2 7 9 3
0
```

Salida de ejemplo

```
diana
felipe gerardo ricardo
```

D
El cumpleaños de George

George Boole se hace mayor. Cada año que pasa aparece una vela extra en su tarta de cumpleaños y, al mismo tiempo, sus pulmones pierden fuelle y tienen cada vez menos fuerza para apagarlas. Siempre tiene la tentación de recurrir a las velas con números, pero no le gustan. Desde joven ha sido muy de *base dos*, y prefiere quedarse sin resuello antes de estropear su tarta con dígitos tan altos.

Pero este año tiene un plan. Para reducir el número de velas, va a poner su nueva edad en binario. Cada vela representará un *bit*, y la vela estará encendida para representar un 1, o apagada para indicar un 0. Gracias a eso, no solo usará muchas menos velas, sino que ¡ni siquiera tendrá que encender todas! Está tan seguro de que podrá por fin apagar todas las velas que hasta está decidido a poner su edad en segundos.

Entrada

El programa leerá, de la entrada estándar, un primer número indicando cuántos casos de prueba vendrán a continuación.

Cada caso de prueba es un número $1 \leq n \leq 2 \cdot 10^9$ con la edad de George *en segundos*.

Salida

Por cada caso de prueba el programa escribirá el número de velas encendidas que tendrá que poner George en su tarta de cumpleaños, sabiendo que representa el número en binario.

Entrada de ejemplo

```
4
1
2
15
8193
```

Salida de ejemplo

```
1
1
4
2
```

E
Keating vs. Pritchard

En su primera clase, el profesor Keating usa el libro de texto del Dr. J. Evans Pritchard de una forma muy particular. Pide a algún alumno que lea los párrafos iniciales del prólogo "Entender la poesía":

> "
> Para entender a fondo la poesía, debemos antes familiarizarnos con su métrica, rima y figuras retóricas, y luego hacernos dos preguntas:
>
> 1. ¿Con cuánto talento se ha conseguido el objetivo del poema?
> 2. ¿Qué importancia tiene dicho objetivo?
>
> La pregunta 1 mide la perfección del poema. La pregunta 2 su importancia. Y una vez estas preguntas están contestadas, determinar la grandeza resulta una tarea relativamente fácil.
>
> "

"Un excremento. Eso me parece el señor Evans Pritchard. Quiero que todos arranquen esa página".

Luego continua pidiendo que arranquen otras páginas con reflexiones sesudas sobre poesía a las que no ve sentido, y los libros de texto de sus estudiantes terminan quedando diezmados.

Entrada

Cada caso de prueba comienza con un número n indicando cuántas páginas pide el profesor Keating a sus alumnos que arranquen. Después aparecen n números, entre 1 y 1.000, con los números de las páginas que deben arrancar.

Se garantiza que no hay números repetidos. El libro del Dr. Pritchard está impreso a doble cara.

La entrada termina con un `0` que no debe procesarse.

Salida

Por cada caso de prueba el programa escribirá el número *de hojas* que son realmente eliminadas.

Entrada de ejemplo

```
1
21
2
21 22
2
22 23
4
30 21 27 22
0
```

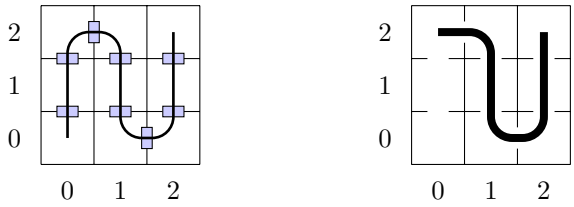
Salida de ejemplo

```
1
1
2
3
```

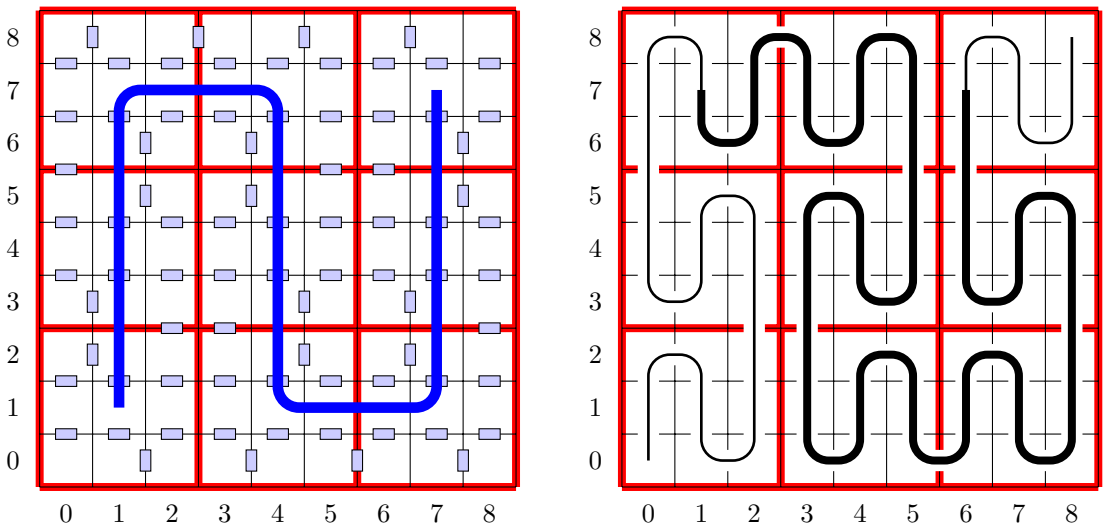
F
La granja de Peano

Cuando Giuseppe Peano nació en 1858 en una granja, nadie podía sospechar que se convertiría en uno de los grandes matemáticos italianos de la época. Cálculo infinitesimal, axiomatización de las matemáticas, lógica matemática... incluso propuso el uso de un lenguaje universal, *interlingua* que ha llegado a nuestros días. Una de sus contribuciones más famosas es la llamada *curva de Peano* con la que mostró que una curva continua puede ser encerrada en una región arbitrariamente pequeña.

En realidad esa curva ya la había utilizado en sus años de adolescencia en la granja familiar. Cuando su padre decidió extender el negocio y comprar vacas, el jóven Giuseppe le propuso colocar los animales de una forma peculiar. Comenzó dividiendo el terreno en parcelas cuadradas (una para cada vaca) organizadas en una especie de rejilla de 3×3. En lugar de conectar todas las parcelas con sus adyacentes abrió las puertas justas para que solo hubiera un camino para ir de cualquier parcela a cualquier otra (el camino completo es una especie de N). El resultado es que para moverse, muchas veces se necesitaba dar un buen rodeo. Por ejemplo para ir desde la (0,2) a la (2,2) hay que atravesar 6 puertas a pesar de que la distancia en línea recta es 2.



Cuando esa distribución se le quedó pequeña (al fin y al cabo solo da para 9 vacas), compró los terrenos al norte y al este triplicando tanto el alto como el ancho del terreno original. Con todo ese nuevo terreno ahora podía tener 3×3 bloques de 3×3 parcelas que organizó para visitarse siguiendo la misma especie de N. Esta vez, eso sí, cada bloque en lugar de ser una parcela para una vaca eran 3×3 parcelas que se recorrían, a su vez, siguiengo la misma especie de N (aunque algunas veces invertida). Con esta disposición para ir desde la celda (1,7) a la (6,7) hay que abrir 51 puertas.



Cuando la granja creció más allá de las 81 vacas que entraban, Peano repitió el mismo proceso. Triplicó el tamaño del terreno tanto a lo ancho como a lo alto y replicó el bloque de 9×9 en 3×3 bloques.

Al final llegó a tener una granja con millones de vacas. Lo peor era trasladarlas de una parcela a otra porque muchas veces el camino se hacía larguísimo.

Entrada

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

Cada caso de prueba consiste en una línea con las coordendas ($x1$, $y1$) origen del camino y las coordenadas ($x2$, $y2$) destino. Las coordenadas estarán entre 0 y 3^{19}–1.

Salida

Por cada caso de prueba se escribirá una única línea indicando cuántas puertas habrá que abrir para ir desde el origen al destino. Gracias al límite en el ancho y el alto del terreno, se garantiza que ese número nunca superará 2^{63}.

Entrada de ejemplo

```
4
0 2 2 2
2 2 0 2
1 7 6 7
7 1 7 6
```

Salida de ejemplo

```
6
6
51
19
```

G
De BMW a DKW

Cuando en sus años mozos, mi padre, Ícaros Troza Gal, ayudaba al abuelo cuidando las ovejas en las tardes de verano, veía en el horizonte pasar a los bólidos por la lejana autopista, y soñaba con tener, algún día, una de aquellas maravillas. Su pasión por los coches le llevó a casarse con mi madre cautivado, más que por cualquier otra cosa, por su nombre: Amor Carros. Cuando se enteró de que iban a ser padres de una niña, solo pudo pensar en un nombre para mí: Mercedes. Mi madre insistía en que mi nombre completo, con los dos apellidos, no auguraba nada nuevo con respecto a mi habilidad conduciendo, pero mi padre no quiso escucharla.

Cuando tuve edad para conducir quiso inculcarme su pasión y me dio las primeras lecciones en su coche recién comprado. No era el bólido con el que él había soñado, pero no lo cambiaría por nada. Tenía por matrícula `2500 BMW`, que coincidía con uno de los modelos que más admiró durante su niñez.

El problema fue que mi madre tenía razón. Eso de conducir no se me dio nada bien, y el primer día rayé una puerta, el segundo día dejé un retrovisor colgando al golpearme con un cubo de basura y el tercer día... prefiero no tener que recordarlo. Al final mi padre tuvo que comprarse otro coche, que resultó tener matrícula `2500 DKW`. Me llamó la atención que el número coincidiera con el del coche anterior y pregunté a mi padre cuántos coches de distancia habría entre las dos matrículas.

Lo que me contestó me quitó las ganas de pedirle que siguiera dándome clases de conducir: "menos que las tortas que te voy a dar si te acercas a mi coche nuevo".

Entrada

La entrada comienza con una línea con un número que indica cuántos casos de prueba vendrán a continuación.

Cada caso de prueba está compuesto por una única línea con dos matrículas separadas por un guión. Una matrícula está compuesta por cuatro dígitos seguidos (tras un espacio) por tres letras del alfabeto inglés, salvo las vocales y la `Q`. El orden de ambas no tiene por qué coincidir con el orden de matriculación.

Salida

Por cada caso de prueba se escribirá el número de vehículos que se han matriculado desde una a la otra, ambas incluidas. La forma en la que se matriculan los coches incrementa los números de uno en uno desde el `0000`. Cuando se llega al `9999` la cuenta comienza de nuevo en `0000` y se cambia la tercera letra para pasar a la siguiente del abecedario inglés saltándose las vocales y la `Q`. Cuando se llega a la `Z` se vuelve a la `B` y se incrementa la letra situada a su izquierda.

Entrada de ejemplo

```
4
2500 BMW - 2500 DKW
0000 BBB - 0000 BBC
0000 BBC - 0000 BBB
0011 CPP - 0017 CPP
```

Salida de ejemplo

```
7600001
10001
10001
7
```

H
Noches de 12 horas

La medición del tiempo no siempre ha sido fácil. Los primeros mecanismos para medir el tiempo fueron relojes de sol (ideados hace más de tres mil años) que, obviamente, no servían de noche. Su funcionamiento es bastante simple: una estaca de madera clavada proyecta una sombra que se va moviendo según avanza el día. Según muchos historiadores fueron los egipcios los que dividieron ese trayecto en doce partes, dando lugar a lo que hoy conocemos como las horas: doce horas de luz y, por simetría, doce de oscuridad.

Esos relojes, no obstante, no eran muy precisos. Debido a la inclinación de la tierra con respecto al sol, es sabido que, a no ser que vivas en el ecuador terrestre, la cantidad de horas de luz y de sombra varía según la época del año. De hecho solo en dos momentos concretos del año (durante los equinoccios) la duración del día y de la noche coinciden.

Eso significa que la duración de cada una de las doce "horas" que mide un reloj solar no es la misma durante todo el año. Como en verano hay más tiempo de luz pero las mismas 12 "horas", estas son más largas que en invierno. Este desajuste no parecía suponer ningún problema para las gentes de la antigüedad; simplemente se habituaron a que el tiempo "cundía más" en verano que en invierno. La situación, de hecho, siguió así hasta aproximadamente el siglo XIV con los primeros relojes mecánicos.

Lo que era realmente difícil, no obstante, era medir el tiempo por la noche. Se sabe que en Egipto (y quizá también en otras civilizaciones anteriores) utilizaban relojes de agua: un recipiente con agua y un pequeño agujero va perdiendo líquido a velocidad constante. Cuando el recipiente se vacía significa que ha transcurrido una hora; en ese momento se llena y el proceso vuelve a empezar.

El problema con ese invento era otro: conseguir que el reloj se vaciara exactamente 12 veces durante la noche (para medir 12 "horas" exactas), independientemente de la época del año... Si el recipiente pierde 48 mililitros de líquido por minuto, ¿con cuánta agua hay que rellenarlo?

Entrada

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

Cada caso de prueba consiste en una única línea con las horas (en formato 24 horas) en las que comienza y termina la noche respectivamente. Se asume que en un día (entre las `0:00` y las `23:59`) la salida de sol siempre ocurre antes que la puesta o lo que es lo mismo, la noche comienza un día y termina el siguiente.

Salida

Por cada caso de prueba se escribirá una única línea indicando el número de mililitros que hay que poner cada vez que se vacía el recipiente (que pierde 48 mililitros por minuto) para que el reloj de agua perciba que la noche dura 12 "horas" exactas.

Entrada de ejemplo

```
2
20:00 8:00
16:30 10:30
```

Salida de ejemplo

```
2880
4320
```

I
¡Soldados! ¡Numérense!

Antes de cada batalla era habitual que los centuriones romanos pidieran a todos sus soldados que se numeraran, para confirmar que no faltaba ninguno que hubiera desertado ante el riesgo inminente de la lucha. El primer soldado decía su número y, a continuación, todos sabían quién debía decir el suyo después.

Este proceso es habitual hoy en otros contextos, pero en la Antigua Roma, dar el número siguiente a otro no era tan fácil como ahora porque los números romanos no eran especialmente cómodos para sumar.

Los números romanos son un *sistema aditivo/sustractivo*, donde se utilizan algunas letras para representar valores, y el uso de varias de ellas suma o resta al acumulado total. La tabla siguiente indica los símbolos válidos en los números romanos y su valor en nuestra notación decimal:

`I`	1	`V`	5	`X`	10	`L`	50	`C`	100	`D`	500	`M`	1000

Las reglas de la notación moderna de números romanos son:

- Los símbolos se escriben de izquierda a derecha, empezando por los de mayor valor.
- Un símbolo seguido por otro de valor igual o menor, *suma* (`XI` es 11), mientras que un símbolo seguido por uno de mayor valor, *resta* (`IX` es 9).
- La unidad (`I`) y demás potencias de 10 (`X`, `C` y `M`) se pueden repetir tres veces seguidas sumando.
- El resto (`V`, `L` y `D`) no pueden repetirse (su suma daría un símbolo de las potencias de 10) ni usarse restando.
- La unidad (`I`) y resto de potencias de 10 pueden aparecer restando, sin repetirse, a los dos símbolos *inmediatamente superiores* (es válido `IV` y `IX` pero no `IL` o `IC`).

Entrada

La entrada está compuesta de un conjunto de líneas, cada una con un número romano (con letras mayúsculas) siguiendo las reglas descritas, entre `I` (1) y `MMMCMXCVIII` (3998).

Salida

Por cada caso de prueba el programa escribirá el valor siguiente al de la entrada, en números romanos en mayúscula siguiendo las reglas de la notación moderna.

Entrada de ejemplo

```
I
III
XXXIX
MCDXCIX
```

Salida de ejemplo

```
II
IV
XL
MD
```

J
Pasos en la escalera

Pepa Taslargas tiene una capacidad envidiable: es capaz de subir varios escalones de una sola vez. Hay gente que puede subir de golpe dos escalones, pero para ella eso es de aficionados. Su habilidad la ha llevado a recorrer el mundo participando en las llamadas *carreras verticales*, como la *carrera de torres* más famosa y antigua del mundo en la que los participantes deben subir los 1576 escalones del Empire State Building.

Una persona normal necesitaría 1576 pasos para subir todos esos escalones. Subiéndolos de dos en dos, son suficientes 788 y de tres en tres solo 526.

Entrada

El primer número de la entrada indica cuántos casos de prueba deberán ser procesados.

A continuación aparece una línea por cada caso de prueba, con dos números entre 1 y 1.000.000. El primero indica el número de escalones de una escalera y el segundo cuántos escalones se pueden subir de una sola vez.

Salida

Por cada caso de prueba se escribirá el número mínimo de pasos necesarios para subir todos los peldaños.

Entrada de ejemplo

```
5
1576 1
1576 2
1576 3
4000 1999
3 4
```

Salida de ejemplo

```
1576
788
526
3
1
```

K
Reconstruyendo la muralla

Villa Rotejos ha presumido durante siglos de la muralla romana que rodea su casco antiguo. Por desgracia, las últimas décadas de presión urbanística han hecho que constructores sin escrúpulos destrocen nuestro patromonio cultural quitando, con nocturnidad y alevosía, piedra a piedra de la parte de arriba de muchas zonas, para utilizarlas en sus nuevas construcciones.

El resultado es que cada sección de la muralla tiene una altura distinta, dependiendo de cuánto haya sido atacada por los vándalos. A nadie se le escapa que la única zona a la que no le falta ni una sola piedra y mantiene su altura original es la que se ve desde la ventana del alcalde.

Afortunadamente en las últimas elecciones ha habido relevo generacional y el nuevo alcalde, Armando Mura Llitas, está decidido a recuperar el antiguo esplendor, igualando la altura de toda la muralla a la de su sección más alta. Desgraciadamente el presupuesto es reducido, y lo irá haciendo poco a poco según vaya consiguiendo los fondos.

Tras anunciarlo en el pregón, el buzón del Ayuntamiento se ha llenado de solicitudes de los vecinos, cada uno pidiendo que se comience la restauración por la parte que se ve desde su ventana.

Entrada

El programa deberá leer, de la entrada estándar, un primer número indicando cuántos casos de prueba tendrán que procesarse. Cada uno comienza con un número $1 \leq N \leq 500.000$ con la longitud, en número de piedras, de la muralla. A continuación aparecen N números con el número de piedras a lo alto que aún quedan en la muralla en cada sección.

Después viene un número Q con el número de solicitudes de los vecinos que han llegado al Ayuntamiento, seguido de una línea por cada una, describiéndola. Todas las solicitudes están compuestas por dos números, $1 \leq A \leq B \leq N$, con el punto de inicio y final de la muralla que el vecino en cuestión quiere que se arregle primero.

Salida

El programa deberá escribir, para cada consulta, el número de piedras que se necesitarían para reconstruir la sección solicitada de modo que alcance la misma altura que la parte más alta de la muralla.

Después de cada caso de prueba se escribirá una línea con tres guiones (`---`).

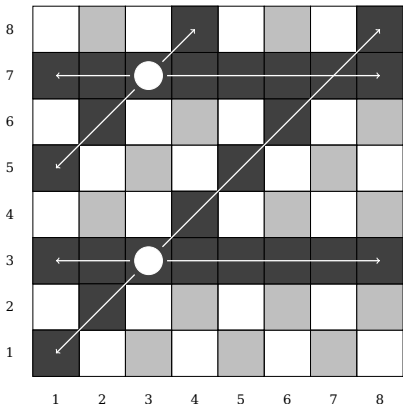
Entrada de ejemplo

```
2
5
3 4 2 5 1
4
1 1
1 2
3 4
3 5
3
0 0 2000000000
1
1 3
```

Salida de ejemplo

```
2
3
3
7
---
4000000000
---
```

L
Torfiles



En el juego del ajedrez hay distintos tipos de piezas cuyas reglas de movimiento varían. Por ejemplo la torre se mueve en horizontal o vertical, mientras que el alfil se mueve en ambas diagonales. Por su parte, la dama puede verse como la "suma" de alfil y torre, pues puede moverse tanto en horizontal y vertical como en cualquier diagonal.

Si entendemos la dama como la suma de alfil y torre, podemos inventarnos una figura nueva, el "*torfil*" que es mitad torre, mitad alfil: se mueve en horizontal pero no en vertical y en una diagonal pero no en las dos. En concreto, un *torfil* situado en la esquina inferior izquierda puede moverse por toda la fila inferior y por toda la diagonal hasta la esquina superior derecha.

Es curioso darse cuenta de que, en un tablero vacío, el número de posiciones a las que puede ir una figura como el *torfil* no siempre es el mismo. Por ejemplo si en lugar de situarlo en la esquina inferior izquierda lo situamos en la esquina inferior derecha su únicos destinos posibles serán la fila inferior.

Dado un tablero de $N \times N$ casillas con una serie de *torfiles* colocados en él, ¿cuántas casillas no son alcanzables por ninguna figura?

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno ocupando varias líneas.

La primera línea de cada caso contiene dos números: el número N que representa el tamaño del tablero, que siempre es cuadrado ($1 \leq N \leq 300.000$) y el número P de *torfiles* que hay sobre él ($0 \leq P \leq min(4 \times N, N \times N)$).

A continuación aparecerán P líneas con las coordenadas de cada uno de los torfiles. El primer número representará la fila y el segundo la columna (la coordenada `1 1` representa la esquina inferior izquierda).

Salida

Por cada caso de prueba se escribirá un único número con la cantidad de casillas del tablero que no son alcanzables, en un solo movimiento, por ningún *torfil*. Las casillas ocupadas por *torfiles* al principio se consideran alcanzables.

Entrada de ejemplo

```
8 2
7 3
3 3
4 0
300000 0
```

Salida de ejemplo

```
39
16
90000000000
```