6
Paréntesis equilibrados

Decimos que una secuencia de caracteres que, entre otros caracteres, contiene paréntesis, llaves y corchetes abiertos y cerrados, está *equilibrada* si de cada uno de ellos tiene tantos abiertos como cerrados y si cada vez que aparece uno cerrado, el último que apareció aún no emparejado fue su correspondiente abierto.

Entrada

La entrada está formada por una serie de casos de prueba, cada uno en una línea. Cada caso consiste en una secuencia de caracteres.

Salida

Para cada caso de prueba se escribirá en una línea la palabra `SI` si la secuencia de caracteres está equilibrada y `NO` en caso contrario.

Entrada de ejemplo

```
Hola, me llamo Jose (Pepe para los amigos).
Yo soy Francisco (o Paco]
{((())}[]
]
```

Salida de ejemplo

```
SI
NO
SI
NO
```

Autor: Alberto Verdejo.

7
El juego de la linterna

Jimmy tiene muchos sobrinos a los que tiene que entretener cada vez que van a su casa de visita. Hace tiempo que encontró la fórmula para conseguirlo. En realidad para él fue una sorpresa descubrir que aquel medio juego improvisado que empezó como una manera de tenerles tranquilos cinco minutos se ha convertido en una tradición que le piden fin de semana tras fin de semana.

El "juego" (por llamarlo de alguna forma) consiste en lo siguiente: Jimmy deja a todos los sobrinos en la habitación. Mientras él espera en el pasillo, los niños se colocan al azar uno al lado del otro formando una hilera de personajitos de distintas edades y alturas. Cuando ya están todos colocados, se apagan las luces y el tío Jimmy entra a oscuras y se coloca frente a la fila. Y entonces es cuando empieza la parte "divertida" del juego.

Jimmy saca una linterna de su bolsillo e ilumina la cara de uno de sus sobrinos al azar, que da un paso adelante. Después se desplaza a la derecha saltándose a un número de sobrinos al azar e ilumina a otro de los niños que avanza también saliéndose de la fila. El proceso lo repite una última vez, dejando a tres niños elegidos aleatoriamente e iluminados por la luz de la "linterna mágica".

En ese momento, se da la luz y los tres sobrinos que han sido elegidos son premiados con cualquier cosa; unas veces son caramelos, otras veces es alguna moneda que los niños gastarán en su próxima visita a la tienda de chucherías...

Eso sí, el tío Jimmy pone una única condición para el premio: cuando los tres sobrinos que han dado un paso adelante salen de la fila, sus alturas son variopintas. Dependiendo de a quién haya elegido, el primero puede ser el más bajito, el más alto o el del centro. Los tres niños pueden, por tanto, quedar en orden creciente de altura, en orden decreciente o de cualquier otra forma. Pues bien, los niños se quedarán *sin premio* si el primer sobrino elegido es el más bajito de los tres, el segundo es el más alto y el tercero es el mediano.

Albertito, el primo mayor, ha decidido dirigir a todos sus primos y en el momento de colocarse supervisa si la colocación final puede llevar o no a que el tío Jimmy elija a tres niños de forma que no haya premio. Y en ese caso les pide a todos que se recoloquen.

Por ejemplo, si ese día hay 5 niños y se colocan aleatoriamente de forma que las alturas quedan `5 1 4 3 2`, el tío Jimmy podría tener la "puntería" de sacar al de altura `1`, al de `4` y al de `2` y se quedarían sin premio. Sin embargo, en una colocación como `5 1 2 3 4` los sobrinos elegidos nunca se quedarán sin premio.

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno ocupando dos líneas. La primera línea contiene un número indicando el número de sobrinos que hay ese día en casa (como mínimo 3 y hasta 500.000). La segunda línea contendrá las alturas de cada uno de ellos en la configuración aleatoria en la que se han colocado. Las alturas no se repiten, y serán números entre 1 y 10^9.

Salida

Para cada caso de prueba se indicará el veredicto que debe dar Albertito ante esa colocación. Se escribirá `SIEMPRE PREMIO` si el tío Jimmy nunca podrá sacar una configuración de sobrinos sin premio, o `ELEGIR OTRA` si el tío puede sacar a tres sobrinos de forma que se queden sin premio.

Entrada de ejemplo

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

Salida de ejemplo

```
ELEGIR OTRA
SIEMPRE PREMIO
```

Autores: Marco Antonio Gómez Martín y Pedro Pablo Gómez Martín.

Problema número 187

Solitario

Tiempo máximo: 3,000-4,000 s Memoria máxima: 4096 KiB

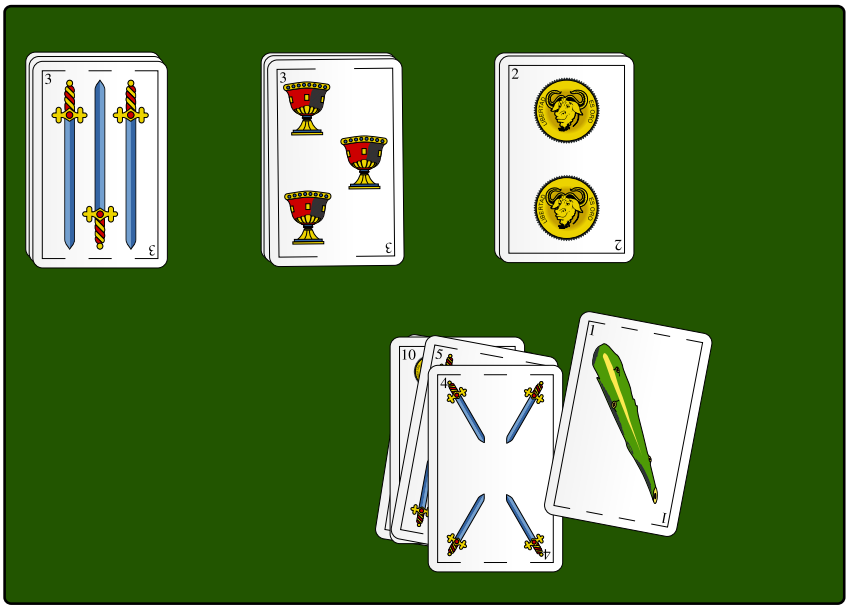
http://www.aceptaelreto.com/problem/statement.php?id=187

Maneras de jugar al solitario hay tantas como tipos de barajas y países donde se juega con ellas. Siendo muy pequeña, mi abuelo materno me enseñó a jugar a una variedad del solitario muy sencilla utilizando la baraja española. Como en la mayoría de los solitarios, el objetivo es crear cuatro pilas de cartas, una para cada palo, en orden ascendente, es decir del as al rey.

En la modalidad de mi abuelo, se baraja el mazo de cartas y se coloca boca abajo sobre la mano. En cada jugada, se cogen simultáneamente *dos* cartas de ese mazo y se dan la vuelta sobre la mesa, formando una pila de cartas visibles. De esa pila, sólo podrá retirarse en cada momento la carta situada más arriba.

Durante la partida, se construyen también cuatro pilas de cartas, una para cada palo. Para eso, es necesario primero tener la suerte de que el as de cada palo quede en la parte de arriba de las cartas según van siendo llevadas desde el mazo de la mano a la mesa. Cuando eso ocurre, el as se retira y se coloca iniciando su propia pila, sobre la que podrá luego colocarse el dos de ese palo, luego el tres, etcétera, siempre que queden visibles en el mazo de cartas descubiertas.

Cuando se retira la carta de la parte superior del mazo de cartas descubiertas, la que queda inmediatamente debajo pasa a quedar visible; si es posible colocarla en alguna de las pilas de los palos que se están construyendo, deberá ser colocada también; el proceso continuará hasta que no queden cartas visibles, o la superior no pueda ser colocada.



Por ejemplo, imagina que, tras varias cartas descubiertas, hemos conseguido iniciar las pilas de los palos espadas, copas y oros colocando varias de sus cartas, llegando a la situación de la figura. En la penúltima jugada, extragimos del mazo de cartas boca abajo que tenemos en la mano la sota de oros y el cinco de espadas, que no pudieron ser colocadas. En el último hemos tenido más suerte al sacar el cuatro de espadas y el as de bastos. Éste queda en la parte superior y podemos colocarlo sobre la mesa, iniciando su propia pila. Además, al retirar el as, queda al descubierto el cuatro de espadas, que también colocamos inmediatamente (sobre el tres), y éste a su vez deja visible al cinco, que también extraemos. La sota de oros, que pasa a ser la superior, no se puede colocar, por lo que llega el momento de probar suerte descubriendo dos cartas nuevas del mazo de la mano.

Es importante darse cuenta de que si en lugar del as de bastos hubiera quedado en la parte superior, por ejemplo, el dos de bastos, al no poderlo retirar no podríamos tampoco colocar el cuatro de espadas de debajo.

Lo habitual es que tras llevar todas las cartas por parejas del mazo boca abajo a la mesa, no hayamos sido capaces de construir completas las cuatro pilas de cartas de cada palo. En ese caso, recogemos, sin barajar, las cartas visibles que no hemos colocado y les damos la vuelta, de modo que la última carta que extragimos se convierte en la inferior del mazo boca abajo. Después, repetimos el proceso sacando las cartas de dos en dos otra vez. Es posible que al terminar de descubrir todas las cartas nos quede una única carta en la mano, y no dos. En ese caso, se colocará esta última sobre la pila de cartas visibles y se comprobará si se puede colocar siguiendo el procedimiento habitual.

El jugador gana la partida si consigue colocar todas las cartas en las pilas de los palos, y pierde en caso contrario, lo que ocurre cuando da toda una vuelta a las cartas pendientes sin haber sido capaz de colocar ninguna.

Entrada

La entrada consta de una serie de casos de prueba terminados con un `0`. Cada caso de prueba consiste en una configuración de baraja formada por uno, dos, tres o cuatro palos; es decir, 10, 20, 30 o 40 cartas. Cada palo consta de 10 cartas con valores desde el `1` (As) hasta el `7`, `10` (sota), `11` (caballo) y `12` (rey).

El caso de prueba comenzará por un número que indica el número de palos con el que jugamos el solitario. En la línea siguiente aparecerá la configuración del mazo de cartas inicial, ya barajado, como una secuencia de cartas. El mazo se considera colocado boca abajo, de manera que la primera carta que aparece será la la primera que tengamos levantar.

Cada carta se representa con un número y un carácter, que indican el valor de la carta y el palo al que pertenece respectivamente. Se utilizará `O` para los oros, `C` para las copas, `E` para las espadas y `B` para los bastos. Entre el número y el palo aparecerá un espacio, al igual que entre una carta y la siguiente.

Salida

Para cada caso de prueba se mostrará una línea en la que se escribirá `GANA` si el jugador será capaz de terminar con éxito el solitario con esa configuración y `PIERDE` en caso contrario.

Entrada de ejemplo

```
1
1 B 2 B 3 B 4 B 5 B 6 B 7 B 10 B 11 B 12 B
1
12 E 2 E 3 E 7 E 11 E 10 E 6 E 5 E 4 E 1 E
2
2 O 1 O 2 C 1 C 4 O 3 O 4 C 3 C 12 O 11 O 12 C 11 C 10 O 7 O 10 C 7 C 6 O 5 O 6 C 5 C
0
```

Salida de ejemplo

```
PIERDE
GANA
GANA
```

Autores: Patricia Díaz García, Pedro Pablo Gómez Martín y Marco Antonio Gómez Martín.

Revisores: Ferran Borrell Micola y Cristina Gómez Alonso.

8
Duplicar una lista enlazada

En este ejercicio se trata de practicar con las estructuras de datos de listas enlazadas (simples o dobles). Se leerá una serie de valores de la entrada que se guardarán en una lista enlazada. A continuación, se recorrerá la lista duplicando todos los nodos (cada nodo se convertirá en dos nodos consecutivos con el mismo valor). Por último, se recorrerá la nueva lista mostrando el valor en todos sus nodos.

Requisitos de implementación.
Para implementar el ejercicio se extenderá una clase que utilice una lista enlazada de nodos (`queue` o `deque`, o mejor primero con una y luego con otra) con dos métodos públicos: uno para duplicar los nodos de la lista y otro para mostrar el contenido de la lista (desde el primer elemento hasta el último).

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en una línea y consiste en una serie de enteros positivos separados por espacios y terminada con un `0` (que no pertenece a la lista). Estos valores se introducirán en una lista enlazada según aparecen de izquierda a derecha.

Salida

Para cada caso de prueba se escribirá en una línea el contenido de la lista duplicada desde el primer elemento hasta el último.

Entrada de ejemplo

```
5 3 1 8 0
0
7 7 0
```

Salida de ejemplo

```
5 5 3 3 1 1 8 8

7 7 7 7
```

Autores: Isabel Pita y Alberto Verdejo.

9

Invertir una lista enlazada

En este ejercicio se trata de practicar con las estructuras de datos de listas enlazadas (simples o dobles). Se leerá una serie de valores de la entrada que se guardarán en una lista enlazada. A continuación, se invertirá el orden de los nodos de manera que el primero se convierta en último, el segundo en penúltimo, etc. Por último, se recorrerá la nueva lista mostrando el valor de todos sus nodos.

Requisitos de implementación.

Para implementar el ejercicio se extenderá una clase que utilice una lista enlazada de nodos (`queue` o `deque`, o mejor primero con una y luego con otra) con dos métodos públicos: uno para invertir los nodos de la lista y otro para mostrar el contenido de la lista (desde el primer elemento hasta el último).

Es importante que la implementación del método que invierte la lista *no reserve nueva memoria dinámica*. Los nodos enlazados deben ser reutilizados.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en una línea y consiste en una serie de enteros positivos separados por espacios y terminada con un `0` (que no pertenece a la lista). Estos valores se introducirán en una lista enlazada según aparecen de izquierda a derecha.

Salida

Para cada caso de prueba se escribirá en una línea el contenido de la lista invertida desde el primer elemento hasta el último.

Entrada de ejemplo

```
1 2 3 4 0
0
8 7 6 0
3 0
```

Salida de ejemplo

```
4 3 2 1

6 7 8
3
```

Autores: Isabel Pita y Alberto Verdejo.

10
Insertar elementos en una lista enlazada

En este ejercicio se trata de practicar las listas enlazadas simples (utilizaremos las vistas en clase para implementar el TAD `queue`, aunque la nueva operación no sea una operación de colas). Queremos una operación que inserte una serie de elementos de una lista enlazada dentro de otra lista enlazada a partir de una posición dada. Por ejemplo, si queremos insertar los elementos 2 4 6 en la lista 1 3 5 7 9 a partir de la posición 2 (las posiciones válidas se numeran de 0 a N, el tamaño de la lista donde se inserta), el resultado sería 1 3 2 4 6 5 7 9.

Requisitos de implementación.

En la resolución del problema, se extenderá *mediante herencia* la clase `queue` con un método que inserte en una lista enlazada los elementos de otra lista recibida como argumento, a partir de la posición indicada. La lista recibida como argumento pasará a ser vacía.

El coste de la operación debe ser lineal con respecto al número de elementos en la lista. No pueden hacerse nuevos `new`s; deben reutilizarse los nodos que ya existen en las listas enlazadas.

También se añadirá un método que permita mostrar el contenido de la lista en una línea, separando sus elementos por espacios.

No modifiques ni subas al juez el fichero `queue_eda.h` cuya clase `queue` debes extender.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en cuatro líneas. La primera contiene el número N de elementos de la lista principal (un número entre 1 y 100.000). En la segunda línea se muestran esos N elementos, números entre 1 y 1.000.000. La tercera línea contiene dos números: el número M de elementos a insertar (un número entre 1 y 100.000); y la posición P de la lista principal donde deben ser insertados ($0 \leq P \leq N$). La cuarta línea contiene los M elementos a insertar, números entre 1 y 1.000.000.

Salida

Para cada caso de prueba se escribirá en una línea la lista modificada tras insertar los elementos a partir de la posición indicada.

Entrada de ejemplo

```
5
1 3 5 7 9
3 2
2 4 6
5
1 3 5 7 9
3 5
2 4 6
5
1 3 5 7 9
3 0
2 4 6
```

Salida de ejemplo

```
1 3 2 4 6 5 7 9
1 3 5 7 9 2 4 6
2 4 6 1 3 5 7 9
```

Autor: Alberto Verdejo.

11
Mezclar listas enlazadas ordenadas

En este ejercicio se trata de practicar las listas enlazadas simples (utilizaremos las vistas en clase para implementar el TAD `queue`, aunque la nueva operación no sea una operación de colas). Queremos una operación que inserte en una lista enlazada ordenada los elementos de otra lista enlazada ordenada recibida como argumento, de tal forma que la lista resultante quede también ordenada. La lista recibida como argumento pasará a ser vacía.

Requisitos de implementación.
En la resolución del problema, se extenderá *mediante herencia* la clase `queue` con un método que implemente esta operación de mezcla ordenada.

El coste de la operación debe ser lineal con respecto a la suma del número de elementos en las listas que se mezclan. No pueden hacerse nuevos `new`s; deben reutilizarse los nodos que ya existen en las dos listas enlazadas.

También se añadirá un método que permita mostrar el contenido de la lista en una línea, separando sus elementos por espacios.

Entrada

La entrada consta de una serie de casos de prueba. La primera línea contiene el número de casos de prueba que vendrán a continuación. Cada caso ocupa dos líneas. Cada una de estas líneas representa una de las listas, y contiene sus elementos ordenados de menor a mayor, una serie de números entre 1 y 1.000.000, seguidos de un 0, que marca el final de la descripción de la lista, sin pertenecer a ella.

Salida

Para cada caso de prueba se escribirá en una línea la lista modificada tras mezclar de forma ordenada los elementos en ambas listas.

Entrada de ejemplo

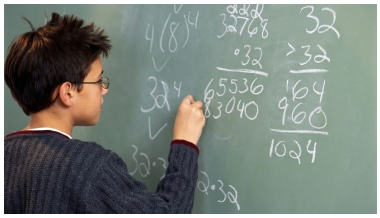
```
6
1 3 5 7 0
2 4 6 8 0
2 4 6 8 0
1 3 5 7 0
1 2 3 0
6 7 8 0
1 10 20 0
2 6 8 12 20 22 0
0
1 2 3 0
1 1 2 2 0
0
```

Salida de ejemplo

```
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 6 7 8
1 2 6 8 10 12 20 20 22
1 2 3
1 1 2 2
```

Autor: Alberto Verdejo.

12
El alumno afortunado



El profesor de EDA ha decidido sacar a la pizarra a un alumno a resolver un problema sorpresa. Para seleccionar al "afortunado" ha numerado a cada uno de los N alumnos con un número del 1 al N y los ha colocado a todos en círculo. Empezando por el número 1, va "salvando" a uno de cada tres (es decir, "salva" al 3, luego al 6, luego al 9, etc.), teniendo en cuenta que al ser circular, cuando llega al final sigue por los que quedan al principio. Siguiendo con el ejemplo, si hubiera 10 alumnos, tras eliminar al 9, se saltaría al 10, se saltaría al 1, y se eliminaría al número 2. El proceso continúa hasta que solamente queda un alumno, que es quién saldrá a la pizarra.

Entrada

La entrada está formada por varios casos de prueba, cada uno en una línea. Cada uno tendrá dos números. El primero indica el número N de alumnos ($1 \leq N \leq 10.000$). El segundo (un número positivo menor o igual a 1.000), indica cuántos alumnos se salta el profesor antes de sacar del círculo a uno de ellos (para el ejemplo anterior este número sería 2). La entrada termina con dos ceros.

Salida

Para cada caso se escribirá en una línea el número del alumno que saldrá a la pizarra.

Entrada de ejemplo

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

Salida de ejemplo

```
4
7
2
```

Autor: Profesores de EDA.

13
La práctica de esteganografía

La *esteganografía* (del griego *steganos*, cubierto u oculto, y *graphos*, escritura) estudia técnicas que permitan ocultar mensajes dentro de otros (llamados portadores) de modo que no se perciba la presencia de los primeros. Es decir, procura ocultar mensajes dentro de otros de modo que el propio acto de la comunicación pase inadvertido para un probable intruso, que ni siquiera sabrá que se está transmitiendo información sensible.

En la escuela de esteganografía de Atenas, a Ocultaniakis le han puesto una práctica: tiene que encontrar la forma de esconder un número en un mensaje. Lo primero que se le ha ocurrido es introducirlo en una lista larga de números, pero le parece que esta solución es muy sencilla y recibirá poca nota en su evaluación. Así que ha decidido dar una vuelta de tuerca más a la esteganografía y hacer que el número oculto no esté presente en el mensaje, sino que haya que calcularlo buscando una clave oculta en el mismo.

La clave estará formada por una serie corta de números distintos. Estos aparecerán, posiblemente varias veces, dentro de una lista larga de números, el mensaje completo. El número secreto será la longitud de la subcadena más corta del mensaje que contenga los números de la clave en el mismo orden. En concreto, si la clave son los números c_1, c_2, ... , c_r, y el mensaje m_1, m_2, ... , m_n, una subcadena del mensaje contiene la clave si existen índices $1 \leq i_1 < \ldots < i_r \leq n$, tales que $m_{i_1} = c_1, \ldots, m_{i_r} = c_r$. Y en ese caso la longitud de la subcadena es $i_r - i_1 + 1$.

¿Sabrías recuperar la información escondida en un mensaje generado por Ocultaniakis?

Entrada

La entrada estará formada por una serie de casos de prueba. Cada caso está formado por 4 líneas: la primera contiene el tamaño R de la clave, entre 2 y 10 números; la segunda contiene los R números de la clave, todos distintos, en el orden en el que deben aparecer en el mensaje; la tercera contiene el tamaño N del mensaje, entre R y 250.000; y la cuarta contiene los N números del mensaje. Se garantiza que la clave siempre aparece al menos una vez oculta en el mensaje. Los números que aparecen tanto en la clave como en el mensaje están entre 1 y 10.000.

Salida

Para cada caso de prueba se escribirá una línea que contenga la longitud (número de elementos) de la subcadena más corta del mensaje que contenga la clave.

Entrada de ejemplo

```
3
1 2 3
10
5 1 3 2 1 7 3 2 3 8
2
3 1
4
31 3 5 1
```

Salida de ejemplo

```
5
3
```

Autor: Alberto Verdejo.

Problema número 143

Tortitas

Tiempo máximo: 1,000-4,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=143

La especialidad de Michelón es hacer tortitas. En la cocina de su restaurante hace a diario muchísimas y las va colocando una encima de otra según las saca de la plancha. Le gusta ver variedad en su pila de tortitas, por lo que va haciéndolas de tamaños distintos.

Como la tortita que está más arriba se enfría mucho más rápido que el resto, de vez en cuando mete la espátula entre dos tortitas de la torre y las da la vuelta, de forma que la tortita que estaba encima queda ahora en medio, y la tortita que estaba en medio, queda la primera.

Tras una serie de volteos, ¿serías capaz de decir el tamaño de la tortita queda más arriba?

Entrada

Cada caso de prueba contiene dos líneas. La primera describe la situación inicial de la pila de tortitas. Contiene una sucesión de números terminada con `-1` que indica los tamaños de cada una de las tortitas, empezando por el tamaño de la situada encima de la mesa y terminando por la de más arriba. Todos los tamaños son positivos, por lo que el `-1` final no debe considerarse.

Una segunda línea describe los volteos que hace Michelón. El primer número de la línea indica el número de volteos que hace. Le sigue un número por cada movimiento, indicando el número de tortitas totales que coge de golpe. Se garantiza que el número siempre es válido (es decir, está entre 0 y el número total de tortitas).

La entrada termina con un caso de prueba sin tortitas.

Salida

Por cada caso de prueba escribe el tamaño de la tortita que queda arriba tras los movimientos de Michelón.

Entrada de ejemplo

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

Salida de ejemplo

```
1
2
5
```

Autor: Marco Antonio Gómez Martín.

Revisor: Pedro Pablo Gómez Martín.

Problema número 198

Evaluando expresiones

Tiempo máximo: 1,000-4,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=198

Las expresiones aritméticas suelen escribirse utilizando lo que se conoce como notación *infija* en la que los operadores se colocan entre los operandos. Esta notación, intuitiva para los humanos, tiene el problema de obligarnos a poner paréntesis en ciertas ocasiones para cambiar el orden de aplicación de los operadores.

Por otro lado, la notación *postfija* consiste en colocar el operador *tras los dos operandos*; una de sus ventajas es que no necesita paréntesis. Además es fácilmente evaluable con una pila. El proceso de evaluación consiste en añadir a la pila los operandos que nos vayamos encontrando. Cuando leemos un operador, extraemos dos valores de la pila los combinamos con el operador encontrado (teniendo en cuenta que el primer valor que se extrae es el segundo operando de la operación) y añadimos el resultado de vuelta.

Existe otra posibilidad de notación que sigue la misma idea que la anterior pero en vez de utilizar una pila para la evaluación, utiliza una *cola*. Cuando se tienen que añadir elementos a la cola, se hace por detrás, mientras que la extracción se realiza por delante.

Dada una expresión, nos preguntamos si, al ser considerada escrita en cada una de las dos notaciones, dará el mismo resultado, uno distinto, o incluso si la expresión no será correcta en alguna de las dos (debido a división por cero).

Entrada

La entrada consta de una serie de casos de prueba, cada uno en una línea distinta. Cada línea debe interpretarse como una expresión a ser evaluada en cada una de las dos notaciones explicadas antes. Los operandos serán siempre dígitos individuales positivos y los operadores podrán ser suma (`+`), resta (`-`), multiplicación (`*`) o división (`/`). Los cálculos deben realizarse con aritmética entera y se garantiza que ningún resultado intermedio excederá 2^{31}-1.

Salida

Para cada caso de prueba se mostrará una línea en la que aparecerá el resultado de evaluar la expresión utilizando una pila y después utilizando una cola. Entre ambos resultados se mostrará el símbolo `=` si ambos coinciden o `!=` si no. Ten en cuenta que algunas de las dos evaluaciones puede fallar; en ese caso se mostrará `ERRROR`.

Entrada de ejemplo

```
2453/*+
6
811-/
11-8/
00/
```

Salida de ejemplo

```
6 != 17
6 = 6
ERROR != -7
0 != ERROR
ERROR = ERROR
```

Autor: Marco Antonio Gómez Martín.

Revisor: Pedro Pablo Gómez Martín.

14
Decodificación de mensajes

El *agente 0069* ha inventado un nuevo método de codificación de mensajes secretos. El mensaje original X se codifica en dos etapas: en primer lugar, X se transforma en X' reemplazando cada sucesión de caracteres consecutivos que no sean vocales por su inversa; en segundo lugar, X' se transforma en la sucesión de caracteres X'' obtenida al ir tomando sucesivamente el primer carácter de X', luego el último, luego el segundo, luego el penúltimo, etc.

Por ejemplo, si X es `Anacleto, agente secreto`, X' sería `Analceto ,agetnes erceto` y X'' (el mensaje cifrado) sería `Aontaelccreet os e,natge`.

¿Sabrías recuperar el mensaje original a partir de un mensaje cifrado?

Entrada

La entrada está formada por una serie de casos de prueba. Cada caso ocupa una línea y representa un mensaje cifrado con el método del agente 0069.

Salida

Para cada caso se escribirá en una línea el mensaje descodificado.

Entrada de ejemplo

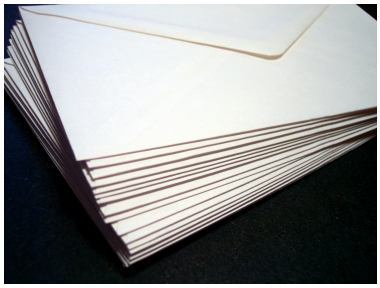
```
Aontaelccreet os e,natge
Moeihr aed eacrepbeadln ied o
AuE ItOoUmbo  ,coortriiuq
```

Salida de ejemplo

```
Anacleto, agente secreto
Me he aprendido el abecedario
AEIOU, borriquito como tu
```

Autor: Alberto Verdejo.

15
Coge el sobre y corre



Pedro Franqueza fue nombrado tesorero hace unos años. Desde entonces sus distintos *chanchullos* le han hecho una persona muy influyente. En la cajonera que tiene debajo de su mesa guarda una hilera de sobres, cada uno con una cantidad de dinero conseguido de manera dudosa.

Hoy tiene una cena de negocios en el bar *Cenás* a la que irá el presidente y otros nueve compañeros. Para mantenerlos contentos, justo antes de salir del despacho mete la mano en el cajón y coge diez sobres consecutivos para repartirlos allí mismo. No ha tenido tiempo de mirar cuánto dinero hay en cada sobre, así que habrá que repartirlos sobre la marcha. Eso sí, el presidente se quedará con el sobre que más dinero tenga.

En el coche yendo hacia el bar va pensando en cómo podría averiguar rápidamente cuánto dinero le correspondería al presidente en base al grupo de 10 sobres consecutivos seleccionados. Y como tiene aires de grandeza, se plantea si sería capaz de hacerlo si guardara hasta 500.000 sobres en su cajonera.

Entrada

La entrada está compuesta por distintos casos de prueba que representan días distintos en la vida de Pedro Franqueza.

Cada caso aparece en dos líneas consecutivas. La primera de ellas contiene dos enteros, el primero con el número de sobres que guarda en el cajón ($1 \leq n \leq 500.000$) y el segundo el número de sobres que tiene que coger para ir a la cena, o lo que es lo mismo, el número de comensales que se llevarán un sobre, incluído el presidente ($1 \leq k \leq n$). La segunda línea contiene n números mayores o iguales a 1 con el dinero que hay en cada uno de los sobres.

La entrada termina con un caso sin sobres, que no debe procesarse.

Salida

Para cada caso de prueba se escribirá una única línea con $n - k + 1$ números que indicarán la cantidad de dinero que se llevará esa noche el presidente dependiendo de qué sobres coja Pedro. En particular, el primer número será el dinero para el presidente si coge los k primeros sobres; el segundo si se salta el primer sobre y coge los k siguientes, etc. En general, el número i-ésimo indica la cantidad que se llevará el presidente si coge los k sobres desde el sobre i hasta el sobre $i + k - 1$.

Entrada de ejemplo

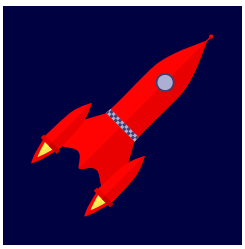
```
6 1
3 8 5 12 15 9
6 3
3 8 5 12 15 9
0 0
```

Salida de ejemplo

```
3 8 5 12 15 9
8 12 15 15
```

Autor: Marco Antonio Gómez Martín.

16
Un viaje a la luna



La empresa *Aterrizaje en la luna* está organizando un nuevo viaje a nuestro satélite. Tiene una lista de todas las personas que se han inscrito en el viaje, con su nombre y su edad. Por razones de seguridad deben excluir a los niños y las personas mayores. Se pide filtrar la lista dejando solo las personas cuya edad se encuentre entre los límites requeridos, incluyendo estos.

Requisitos de implementación.

Implementar un TAD `persona` que tenga como mínimo operaciones para consultar el nombre y la edad de una persona.

Extender la clase `list` vista en clase con un método público que filtre la lista eliminando los elementos que cumplan cierta propiedad.

```
template <class Predicate>
void remove_if(Predicate pred);
```

Esta operación se utilizará para eliminar los viajeros que no cumplan el requisito de edad. Pero la propiedad por la cual se filtra la lista es genérica, de forma que si en algún momento se cambiase el requisito para aceptar viajeros, por ejemplo admitiendo solo aquellos cuyo nombre empieza por vocal, la función `remove_if` no se tendría que modificar.

Utilizar un *objeto función* o una *lambda abstracción* para llamar al método de la clase.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con el número $N > 0$ de personas inscritas al viaje seguido de la edad mínima y máxima requeridas para ir a la excursión. En las N líneas siguientes se muestran los datos de las personas que desean ir a la excursión. Primero se muestra la edad y a continuación separado por un espacio en blanco el nombre y apellidos de la persona. La entrada termina con tres ceros.

Se garantiza que $0 \leq$ *edad mínima* $\leq$ *edad máxima* ≤ 100.

Salida

Para cada caso de prueba se escribirá una línea por cada persona que pueda realizar el viaje, con su nombre. El orden debe ser el mismo que se utilizó en la entrada de datos. Cada caso de prueba acaba con una línea con tres guiones (`---`).

Entrada de ejemplo

```
4 18 60
25 Esteban Garcia
16 Beatriz Castro
35 Rodrigo R. Rodriguez
75 Ana Perez
2 25 40
23 Gonzalo Perez
50 Mercedes Estebez
2 15 50
30 Daniel Gonzalez
15 Gamusino Inquieto
0 0 0
```

Salida de ejemplo

```
Esteban Garcia
Rodrigo R. Rodriguez
---
---
Daniel Gonzalez
Gamusino Inquieto
---
```

Autores: Isabel Pita y Alberto Verdejo.

Problema número 144

Teclado estropeado

Tiempo máximo: 1,000-6,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=144

Ya no se fabrican los teclados como antes. Después de unos pocos meses, a Ramiro el suyo le ha empezado a hacer cosas raras. En concreto, cuando pulsa ciertas teclas el teclado parece interpretar que se han pulsado otras.

Después de un rato de análisis ha llegado a la conclusión de que la pulsación de la tecla del guión (`-`) tiene el mismo resultado que si pulsa la tecla `Inicio`, y el cursor se le va al principio de la línea. Algo parecido le ocurre con el `+`, que se lleva el cursor al final igual que la tecla `Fin`. Y al pulsar `*` se consigue el mismo efecto que con la tecla de la flecha derecha. Pero lo peor de todo es lo que ocurre con el `3`: ¡hace lo mismo que la tecla `Supr`, que borra la letra que hay a la derecha del cursor!

El resultado es que cuando Ramiro se pone a copiar un texto sin mirar la pantalla, el resultado final de lo que ha escrito puede terminar siendo muy distinto de lo que quería escribir. ¿Qué texto saldrá como resultado de la pulsación de una serie de teclas?

Entrada

La entrada consta de un número indeterminado de líneas que deben ser consideradas independientes. Cada una contiene la secuencia de pulsaciones de Ramiro.

Salida

Para cada caso de prueba se mostrará en una línea el texto final que obtendrá Ramiro tras pulsar las teclas indicadas en el orden dado.

Entrada de ejemplo

```
EDA
EDA-333
dD-3*A-E+
EDA-3E*3A
```

Salida de ejemplo

```
EDA

EDA
EDA
```

Autor: Marco Antonio Gómez Martín.

Revisor: Pedro Pablo Gómez Martín.