

Hulk

<http://codeforces.com/problemset/problem/705/A>

Dr. Bruce Banner odia a sus enemigos. Como todos sabemos, él apenas puede hablar cuando se transforma en el increíble Hulk. Es por esto que te pidió ayuda para poder expresar sus sentimientos.



A Hulk le encanta El Origen, y sus sentimientos, al igual que la película, son complicados. Estos tienen n capas. La primera capa es odio, la segunda es amor, la tercera es odio y así sucesivamente...

Por ejemplo si $n = 1$, entonces su sentimiento es “I hate it” o si $n = 2$ es “I hate that I love it”, y si $n = 3$ es “I hate that I love that I hate it” y así.

Por favor ayuda al Dr. Banner.

Input

Una única línea que contiene un único entero n ($1 \leq n \leq 100$) - el número de capas de amor y odio.

Output

Imprime el sentimiento del Dr. Banner en una línea.

Entrada	Salida
1	I hate it

Entrada	Salida
2	I hate that I love it

Entrada	Salida
3	I hate that I love it that I hate it

Moda en Berland

<http://codeforces.com/problemset/problem/691/A>

De acuerdo a las reglas de la moda de Berland, una chaqueta debe tener todos los botones abrochados excepto uno, pero no necesariamente debe ser el último. Además si una chaqueta tiene sólo un botón, éste debe estar abrochado, de modo que la chaqueta no quede abierta.

Dada una chaqueta con n botones. Determina si está abrochada de la forma correcta.



Input

La primera línea contiene un entero n ($1 \leq n \leq 1000$), el número de botones de la chaqueta. La segunda línea contiene n enteros a_i ($0 \leq a_i \leq 1$). Si $a_i = 0$ quiere decir que el i -ésimo botón está desabrochado. En caso contrario $a_i = 1$.

Output

En una única línea imprime “YES” si la chaqueta está abrochada de la forma correcta. En caso contrario imprime “NO”.

Entrada	Salida
3 1 0 1	YES

Entrada	Salida
3 1 0 0	NO

Oso y Encontrando Criminales

<http://codeforces.com/problemset/problem/680/B>

Hay n ciudades en Osolandia, enumeradas desde el 1 al n . Las ciudades están ordenadas en una fila larga. Las distancias entre dos ciudades i y j es igual a $|i - j|$.

Limak es un oficial de policía. Él vive en la ciudad a . Su trabajo es capturar criminales. Lo cual es difícil porque no sabe en qué ciudades hay criminales. Sin embargo, él sabe que a lo más hay un criminal en cada ciudad.

Limak usará el BCD (por las siglas en inglés Bear Criminal Detector: Detector de Criminales Oso). El BCD le dirá a Limak cuántos criminales hay para cada distancia desde una ciudad a . Después de tener esos datos, Limak puede atrapar un criminal en cada ciudad para la cual él está seguro que debe haber un criminal.

Tú sabes en qué ciudades están los criminales. Calcula cuántos atraparé Limak después de usar el BCD.



Input

La primera línea de la entrada contiene dos enteros n y a ($1 \leq a \leq n \leq 100$) - el número de ciudades y el índice de la ciudad en la que vive Limak. La segunda línea contiene n enteros t_1, t_2, \dots, t_n ($0 \leq t_i \leq 1$). Hay t_i criminales en la i -ésima ciudad.

Output

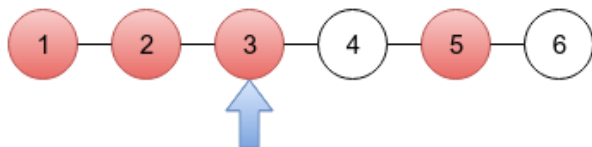
Imprimir el número de criminales que capturará Limak.

Entrada	Salida
6 3 1 1 1 0 1 0	3

Entrada	Salida
5 2 0 0 0 1 0	1

Nota

En el primer ejemplo, hay seis ciudades y Limak vive en la tercera (donde está la flecha azul en la figura de abajo). Las ciudades con criminales están marcadas con rojo.

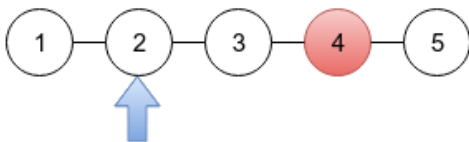


Usando el BCD, Limak tiene la siguiente información:

- Hay un criminal a distancia 0 de la tercera ciudad - Limak está seguro que ese criminal está exactamente en la tercera ciudad.
- Hay un criminal a distancia 1 desde la tercera ciudad - Limak no sabe si ese criminal está en la segunda o en la cuarta ciudad.
- Hay dos criminales a distancia 2 desde la tercera ciudad - Limak está seguro que hay un criminal en la primera y otro en la quinta ciudad.
- Hay cero criminales a distancia mayor.

Así, Limak atraparà a los criminales en las ciudades 1, 3 y 5, lo que hace un total de 3 criminales.

En el segundo ejemplo (figura de abajo), el BCD le da a Limak la información de que hay un criminal a distancia 2 desde la ciudad donde vive Limak. Hay solo una ciudad a distancia 2, así que Limak sabe dónde está el criminal.



Problema A

Comida

archivo: Comida{.c,.cpp,.pas}

Usted ha salido a comer a un restaurant muy fino en donde personas de la farándula y famosos como *Knuth*, *Dijkstra* y *Tarjan* han sido vistos frecuentemente disfrutando de su cena. El restaurante tiene una variedad de 12 opciones y su menú se ve de la siguiente forma:

Nombre del Plato	Precio
Bit de Agua	1
Bit Frito	2
Sopa de Bits	4
Huevo de Bits	8
Byte de Arroz Pequeño	16
Byte de Arroz Mediano	32
Byte de Arroz Grande	64
Byte de Arroz con Curry	128
Lomo de Bits y Bytes	256
Compilado de Assembler	512
Compilado de ANSI-C	1024
Compilado de C++	2048

Note que el i -ésimo plato del menú tiene precio 2^{i-1} (para $1 \leq i \leq 12$). Por ejemplo, el “Byte de Arroz Pequeño”, que es el 5to plato, tiene precio $2^{5-1} = 2^4 = 16$. No todos los días se ofrecen todos los platos, por lo que al llegar al restaurant se le indica a los comensales hasta qué plato del menú se puede elegir ese día.

La única regla de este fino y famoso restaurant es que dependiendo de cuanto dinero usted piensa gastar, siempre se debe pedir la mínima cantidad de platos tal que el costo total sea exactamente la cantidad de dinero que se piensa gastar. Por ejemplo, si todos los platos están disponibles y se piensa gastar 10 pesos en comida, en un restaurant normal usted podría elegir, entre otras, alguna de las siguientes opciones:

- 1) 10 Bits de agua ($1+1+1+1+1+1+1+1+1+1=10$) (10 platos)
- 2) 5 Bits Fritos ($2+2+2+2+2=10$) (5 platos)
- 3) 2 Sopas de Bits con 1 Bit frito ($4+4+2=10$) (3 platos)
- 4) 1 Huevo de bits acompañado con 1 Bits Fritos ($8+2=10$) (2 platos)

Para este restaurant, usted sólo puede elegir la última opción ya que es la combinación de platos mínima (2 platos) para que el costo total sea 10. Ahora, si por ejemplo ese día se está sirviendo sólo hasta el tercer plato (“Sopa de Bits”), entonces sólo puede elegir la opción (3) de arriba que minimiza la cantidad de platos (3 platos).

Entrada

La primera línea contiene un número entero N que corresponde a la máxima cantidad de platos que el restaurant estará ofreciendo (por ejemplo, si $N = 3$, sólo los platos que tienen costo 1, 2 y 4 se servirán). En la siguiente línea encontrará un valor P que corresponde al monto que se piensa gastar.

Salida

Debe imprimir la mínima cantidad de platos de entre los N que se ofrecen ese día, cuyo precio total sea exactamente P .

Subtareas y Puntuación

20 puntos Se probará un conjunto de casos donde el valor de N será siempre 2, mientras que P podrá tomar cualquier valor que cumpla con la condición $1 \leq P \leq 10^8$.

20 puntos Se probará un conjunto de casos donde el valor de N estará entre $2 < N \leq 12$ y el valor de P será siempre una potencia de 2 con la condición que $1 \leq P \leq 2^{N-1}$.

30 puntos Se probará un conjunto de casos donde el valor de N estará entre $2 < N \leq 12$ y el valor de P puede ser un número arbitrario que cumpla con la condición que $P \leq 2^{N-1}$.

30 puntos Se probará un conjunto de casos donde el valor de N estará entre $2 < N \leq 12$ y el valor de P puede ser un número cualquiera tal que $1 \leq P \leq 10^8$.

Entrada de ejemplo	Salida para la entrada de ejemplo
2 11	5

Entrada de ejemplo	Salida para la entrada de ejemplo
10 128	1

Entrada de ejemplo	Salida para la entrada de ejemplo
3 10	3

Entrada de ejemplo	Salida para la entrada de ejemplo
12 111111	58

Comida

Entrada	Salida
2 183234	91617

Entrada	Salida
6 32	1

Entrada	Salida
12 723	6

Entrada	Salida
12 999999	495

Problema D

Git

archivo: git{.c,.cpp,.pas}

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario.

Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

Uno de los tantos comandos que pueden ser ejecutados en un repositorio de Git es *merge*. Este comando permite mezclar dos *branches* o *ramas*. La sintaxis del comando merge es:

```
git merge master devel
```

El ejemplo anterior mezcla la rama *devel* en la rama *master*. Si no hay conflictos o problemas durante el proceso una posible salida de la ejecución sería:

```
Merge made by the 'recursive' strategy.
app/async_tasks/send_alerts_to_pms_async_task.php      | 2 +-
app/libs/payment/paylink_payments/ar_profile_paylink_payment.php | 18 ++++++-----
app/libs/payment/paylink_payments/co_profile_paylink_payment.php | 17 ++++++-----
app/libs/payment/paylink_payments/mx_profile_paylink_payment.php | 2 +-
app/models/invoices_sent.php                             | 10 +++++-----
app/models/s3file.php                                   | 17 ++++++-----
```

La idea de un grupo de estudiantes de una prestigiosa universidad es hacer su propia versión simplificada de Git y tu debes ayudarlos a lograr su objetivo. Una funcionalidad básica es contar el número de inserciones y eliminaciones hechas durante el proceso de mezcla, representadas en el output de Git por los símbolos '+' y '-', para poder mostrar el resumen al final del proceso. En el ejemplo anterior ese resumen sería:

```
6 files changed, 40 insertions(+), 26 deletions(-)
```

En la versión de Git de los estudiantes, el proceso funciona a la inversa. Dadas la cantidad de inserciones y eliminaciones, tu tarea es imprimir la representación textual del resultado de la operación de mezcla usando sólo los símbolos '+' y '-'.

Entrada

La entrada contiene dos enteros I y D ($0 \leq I, D \leq 10^3$) separados por un espacio en blanco que representan el número de inserciones y eliminaciones efectuadas.

Salida

El programa debe imprimir una línea con la representación visual, usando '+' y '-', de la cantidad de inserciones y eliminaciones efectuadas y un salto de línea final. Se debe tener en cuenta que los símbolos representando las inserciones ('+') deben imprimirse antes que los que representan las eliminaciones ('-'), tal como se muestra en los casos de ejemplo de más abajo.

Subtareas y Puntuación

30 puntos Se probará un conjunto de casos de prueba donde $0 \leq I \leq 2$ y $D = 0$.

30 puntos Se probará un conjunto de casos de prueba donde $0 \leq I \leq 10^3$, $D = 0$.

40 puntos Se probará un conjunto de casos de prueba donde $0 \leq I, D \leq 10^3$.

Entrada de ejemplo 1 0	Salida para la entrada de ejemplo +
Entrada de ejemplo 4 0	Salida para la entrada de ejemplo ++++
Entrada de ejemplo 2 3	Salida para la entrada de ejemplo ++---

Git

Entrada	Salida
2 0	++

Entrada	Salida
42 0	++++ ++++

[illegible]

Problema M

Maíz

archivo: `maiz{.c,.cpp,.pas}`

En la granja de Don Pepe las gallinas se alimentan con maíz por una serie de tubos, un tubo por gallina. Cuando Don Pepe quiere alimentar a las gallinas, lanza maíz sobre los tubos y estos empiezan a apilar el maíz, exactamente un grano sobre otro. Suponga que los tubos están numerados de 0 hasta $N - 1$. Cada vez que Don Pepe lanza maíz, su lanzamiento cubre una secuencia contigua de tubos. Por ejemplo si lanza entre el tubo 5 y el tubo 8, cubre los tubos 5, 6, 7, y 8. El efecto del lanzamiento es que cada uno de los tubos alcanzados apila exactamente un grano de maíz adicional al que tenía previamente.

Por ejemplo, suponga que hay 9 tubos, todos inicialmente vacíos, y Don Pepe en su primer lanzamiento cubre entre el tubo 2 y el tubo 6, en su segundo lanzamiento cubre entre 5 y el 8, y en su tercer lanzamiento entre el 3 y el 7. El estado de los tubos después de cada lanzamiento se muestra en la siguiente figura.



Note que, después de los tres lanzamientos, la cantidad acumulada de granos entre los tubos 3 y 8 es 13, y la cantidad acumulada entre los tubos 0 y 4 es 5. Su tarea consiste en, dada una secuencia de lanzamientos de granos de maíz, responder varias preguntas acerca de la cantidad de granos acumulados entre pares de tubos.

Entrada

La primera línea del input contiene un entero positivo N correspondiente a la cantidad de tubos, y la segunda línea un entero positivo L correspondiente a la cantidad de lanzamientos realizados por Don Pepe. Después de esto encontrará L líneas, cada una con un par de enteros X e Y (tales que $0 \leq X \leq Y < N$) correspondiente a los tubos alcanzados por cada uno de los lanzamientos de maíz de Don Pepe. Posteriormente encontrará otro entero positivo M que corresponde a la cantidad de preguntas que debe responder. Después de esto encontrará M líneas, cada una con un par de enteros I y J (tales que $0 \leq I \leq J < N$), correspondiente a los tubos de la pregunta.

Salida

Para cada una de las M preguntas, usted debe responder la cantidad acumulada de maíz entre los tubos I y J consultados. Las respuestas deben seguir el orden de las preguntas y estar cada una en una línea diferente de la salida.

Puntuación

20 puntos Se probará un conjunto de casos donde $N \leq 100$ y $L = M = 1$, es decir habrá sólo un lanzamiento y sólo una consulta, y la consulta será tal que $I = J$ (es decir, se pedirá el valor acumulado en un único tubo).

20 puntos Se probará un conjunto de casos donde $N \leq 100$ y $L = M = 1$, es decir habrá solo un lanzamiento y sólo una consulta (pero la consulta podrá cumplir con $I \neq J$).

20 puntos Se probará un conjunto de casos donde $N, L, M \leq 10.000$.

20 puntos Se probará un conjunto de casos donde $N, M \leq 100.000$ y $L \leq 10.000$.

20 puntos Se probará un conjunto de casos donde $N, L, M \leq 100.000$.

NOTA: para los últimos casos la cantidad de maíz acumulada puede ser mucha por lo que se recomienda usar `long long` (`LongInt` en Pascal) para las variables de respuesta.

Entrada de ejemplo	Salida para la entrada de ejemplo
100 1 21 59 1 35 35	1

Entrada de ejemplo	Salida para la entrada de ejemplo
100 1 20 50 1 10 30	11

Entrada de ejemplo	Salida para la entrada de ejemplo
9 3 2 6 5 8 3 7 2 3 8 0 4	13 5

Maíz

Entrada	Salida
100 1 30 31 1 15 15	0

Entrada	Salida
100 1 30 60 1 10 70	31

Entrada	Salida
100 1 20 60 1 35 36	2

Problema R

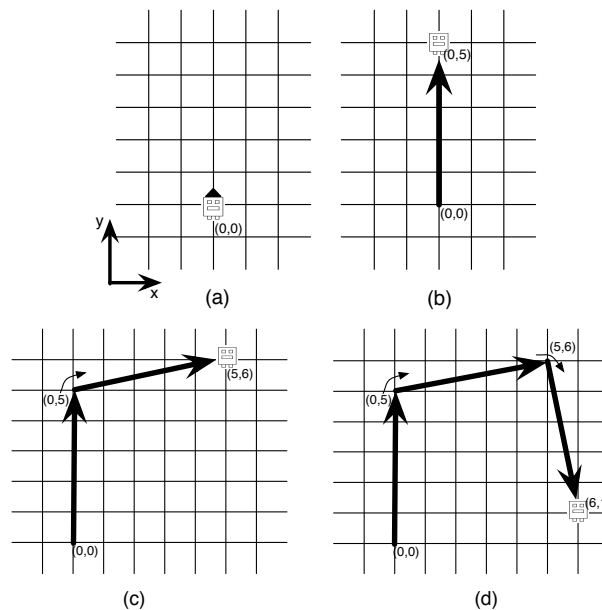
Robot

archivo: robot{.c,.cpp,.pas}

Arturito es el último robot diseñado por la OCI y ha sido entrenado para seguir un plan de navegación formado por puntos en el plano. Arturito es un robot muy curioso, y no quiere obedecer respecto a su plan de navegación hasta que alguien le diga cuántas veces en su camino debe girar a la izquierda y cuántas veces debe girar a la derecha.

La OCI, desesperada por poder convencer a Arturito le pide ayuda a usted. Dado una lista de puntos en el plano que describen el plan de navegación, Arturito comienza en el primer punto de su plan y mirando hacia el segundo punto. Desde ahí en adelante, siempre va de cada punto al siguiente en línea recta. Antes de cada desplazamiento, Arturito debe girar de manera de estar mirando siempre en la dirección en que debe avanzar. Para este comportamiento, usted debe determinar cuántas veces Arturito tendrá que girar a la izquierda y cuántas veces tendrá que girar a la derecha.

Por ejemplo, suponga que Arturito tiene un plan de navegación compuesto por los puntos $(0,0)$, $(0,5)$, $(5,6)$, y $(6,1)$. De acuerdo a esto, Arturito empieza en el punto $(0,0)$ y mirando hacia el punto $(0,5)$. La siguiente secuencia muestra los desplazamientos de Arturito.



En la secuencia se puede observar que Arturito realiza dos giros a la derecha y ninguno a la izquierda. Note que en cada ocasión que Arturito debe girar, escoge el ángulo de giro más pequeño posible. En el último punto de su recorrido Arturito no necesita girar, simplemente se detiene. Note además que Arturito no necesariamente gira en cada punto. Por ejemplo, si en su trayecto debe ir del punto $(0,0)$ al $(0,1)$ y luego al $(0,2)$, dado que estos puntos son *colineales*, Arturito no necesita hacer un giro en el punto $(0,1)$. Puede suponer que el plan de navegación no contiene puntos repetidos, y es tal que Arturito no debe efectuar giros en 180 grados.

Entrada

La entrada contiene un número entero N ($1 \leq N \leq 10^6$) en la primera línea. A continuación siguen N líneas, cada una con dos números enteros separados por un espacio que representan las coordenadas X e Y ($0 \leq X, Y \leq 10^6$) de un punto en el plan de navegación de Arturito.

Salida

Debe imprimir dos números I e D separados por un espacio, y un salto de línea al final, donde I es la cantidad de giros a la izquierda y D es la cantidad de giros a la derecha que debe realizar Arturito en su recorrido.

Subtareas y Puntuación

20 puntos Se probará un conjunto de casos en donde Arturito siempre avanza paralelo a uno de los ejes coordenados (todos los giros en 90 grados), siempre gira a la derecha, y en donde no hay tres puntos seguidos colineales.

20 puntos Se probará un conjunto de casos en donde Arturito siempre avanza paralelo a uno de los ejes coordenados (todos los giros en 90 grados), siempre debe girar a la izquierda, o siempre debe girar a la derecha, y en donde no hay tres puntos seguidos colineales.

20 puntos Se probará un conjunto de casos en donde Arturito siempre avanza paralelo a uno de los ejes coordenados (todos los giros en 90 grados) pero algunos giros podrán ser a la izquierda y algunos a la derecha, y en donde no hay tres puntos seguidos colineales.

40 puntos Se probará un conjunto de casos en donde Arturito puede girar en cualquier dirección y en cualquier ángulo, y en algunos puntos puede que incluso no necesite girar ni a la izquierda ni a la derecha (puntos colineales).

Entrada de ejemplo	Salida para la entrada de ejemplo
4 0 0 0 1 1 1 1 0	0 2

Entrada de ejemplo	Salida para la entrada de ejemplo
5 1 1 0 0 2 0 2 2 3 2	2 1

Robot

¬_ (ツ) _ /¬