Final Caribeña (Eliminatoria) ICPC 2022

Conjunto de problemas del Concurso Real

Autores y desarrolladores de los problemas

Alberto González Rosales - Volvo Cars
Alejandro Jiménez Fabián - Beroly S.A.
Anier Velasco Sotomayor - Harbour Space University
Carlos Joa Fong - Orange Dominicana
Daniel Enrique Cordovés Borroto - Harbour Space University
Ernesto Teruel Velazco - Universidad de las Ciencias Informáticas
Javier E. Forte Reyes - Universidad de las Ciencias Informáticas
Marcelo J. Fornet Fornés - Aurora Labs
Mariano Jason Rodriguez Cisnero - Universidad de Oriente
Reynaldo Gil Pons - University of Luxembourg
Roberto Carlos Abreu Díaz - Meta
Rubén Alcolea Núñez - Universidad de las Ciencias Informáticas

Febrero 18, 2023

Problem A. Again? Solving Queries?

Dado un arreglo de tamaño n, cuyos valores iniciales son todos cero (0), se deben ejecutar tres tipos de consultas:

- *1 i j v*: Incrementar el valor de los números entre los índices i y j (inclusive) en v. Se garantiza que $1 \leq i \leq j \leq n$ y $1 \leq v \leq 1000$.
- *2 i j*: Remplazar los números entre los índices i y j (inclusive) por la secuencia creciente $[1, 2, 3, \ldots, j - i + 1]$. Se garantiza que $1 \leq i \leq j \leq n$.
- *3 i j*: Contar cuántos números entre los índices i y j (inclusive) son divisibles por 5. Se garantiza que $1 \leq i \leq j \leq n$.

Input

La primera línea de la entrada contiene dos enteros $1 \leq n \leq 10^5$ y $1 \leq q \leq 50000$, que denotan el tamaño del arreglo y el número de consultas a realizar, respectivamente. Las siguientes q líneas contienen la descripción de las consultas, en el formato especificado anteriormente. Se garantiza que todos los elementos de entrada son enteros.

Output

Para cada consulta de tipo 3, imprima el resultado en una línea independiente.

Example

standard input	standard output
10 6 1 1 10 5 3 1 10 2 1 5 2 6 10 2 3 8 3 1 10	10 2
5 6 2 1 5 1 1 3 4 1 2 4 1 3 1 5 3 2 5 3 2 4	3 2 1

Problem B. Beyond Connectivity

Se tiene un grafo G, no conexo, de N vértices y M aristas. Se dice que un grafo es no conexo si es posible encontrar un par de vértices tal que ningún camino en el grafo tenga estos vértices como extremos. Se quiere responder Q preguntas. En cada pregunta se tienen dos enteros a y b, y se debe calcular la longitud (en aristas) del camino más corto entre los vértices a y b, en el grafo complemento de G. El complemento de un grafo G es otro grafo G' que no contiene ninguna de las aristas presentes en G y contiene todas las aristas no presentes en G.

Input

La primera línea contiene dos enteros $1 \leq N, M \leq 10^5$. La cantidad de vértices y aristas del grafo, respectivamente. Las siguientes M líneas contienen dos enteros $1 \leq a, b \leq N, a \neq b$. Cada línea representa una arista bidireccional en el grafo G. La siguiente línea contiene un entero $1 \leq Q \leq 10^5$, representando la cantidad de preguntas a responder. Las siguientes Q líneas contienen dos enteros $1 \leq a, b \leq N, a \neq b$. Cada línea representa una pregunta, en el formato descrito anteriormente.

Output

Para cada una de las Q preguntas, se debe imprimir un entero representando la longitud (en aristas) del camino más corto entre los vértices a y b en el grafo complemento de G.

Example

standard input	standard output
5 3 1 2 2 3 4 5 4 1 2 2 3 3 4 4 5	2 2 1 2

Problem C. Computing Near-Death Experience

Debido a la edad del Prof. Farnsworth, finalmente se vio obligado a mudarse a la Estrella de la Muerte Cercana, un satélite artificial cuyos habitantes ancianos están conectados a cápsulas que generan energía y mantienen las mentes de los habitantes entretenidas en una simulación de realidad virtual. Como su casi última voluntad, el Prof. Farnsworth ha elegido "vivir" en una simulación de la granja en la que vivió cuando era niño. Allí tiene mucho espacio para hacer e inventar cualquier locura que quiera. De hecho, una de sus últimas invenciones es de la que más orgulloso está. Consiste en una máquina que dispara círculos sobre un lienzo enorme. Bueno, está bien, no es gran cosa, pero lo es para él. De hecho, es muy entretenido para él intentar cubrir todos los N agujeros ubicados al azar en el lienzo cada día por la mañana utilizando la cantidad mínima de disparos. La máquina solo puede disparar círculos de un radio determinado r sobre el lienzo. Nuestro profesor quiere algo (tal vez una aplicación) que pueda ayudarlo a determinar si logró su objetivo o no, diciéndole cuál es la cantidad mínima K de disparos necesarios para cubrir todos los agujeros.

Input

La primera línea contiene un entero N y un número real r ($1 \leq N \leq 20$, $1 \leq r \leq 2000$), la cantidad de puntos y el radio, respectivamente. Luego siguen N líneas, cada una de las cuales tiene las coordenadas x_i y y_i ($-1000 \leq x_i, y_i \leq 1000$) de un agujero.

Output

Un número K, la cantidad mínima de disparos necesarios para cubrir todos los agujeros.

Example

standard input	standard output
6 2 0 0 2 1 2 -1 4 1 4 -1 6 0	2

Problem D. Domino

En un juego de dominó cada ficha tiene dos números entre 0 y 6. Las fichas se colocan boca abajo y luego se barajan. Antes de empezar el juego, cada jugador selecciona siete fichas. El primer jugador selecciona una ficha cualquiera de su mano y la pone boca arriba. El próximo jugador debe seleccionar una ficha que coincida con uno de los números en la ficha jugada anteriormente. A continuación, el próximo jugador debe seleccionar una ficha que coincida con uno de los extremos disponibles en el tablero, es decir, los números para los cuales no se ha colocado una ficha (ambos números pueden ser iguales). Si un jugador no puede jugar una ficha, este cede su turno al próximo jugador. El juego termina cuando se juegan todas las fichas. El jugador que coloca la última ficha en el tablero gana el juego. Fito ha estado jugando algún tiempo hasta que se queda con una sola ficha. Él no está seguro si puede ganar el juego o no y necesita tu ayuda. Dada la última ficha de Fito y los extremos disponibles del tablero, determine si Fito puede ganar el juego o no.

Input

La primera línea contiene cuatro números. Los dos primeros números corresponden a la última ficha de Fito ($0 \leq A, B \leq 6$). Los dos números siguientes representan los extremos disponibles del tablero ($0 \leq C, D \leq 6$).

Output

Imprima **YES** si Fito puede ganar el juego, **NO** en caso contrario.

Example

standard input	standard output
1 2 3 4	NO
1 2 2 1	YES

Problem E. Enjoy the Chat Group

Estás trabajando en una nueva funcionalidad para tu popular aplicación de mensajería, la cual consiste en notificar si un mensaje fue leído. La idea es mostrar, debajo de cada mensaje, un ícono con la foto de perfil de cada usuario que leyó dicho mensaje. Como se espera que eventualmente cada usuario leerá todos los mensajes, tu equipo decidió imponer las siguientes reglas para esta funcionalidad:

1. No pueden haber íconos duplicados debajo de un mensaje en particular.
2. Si un usuario leyó multiples mensajes, su ícono aparecerá en el último mensaje (en orden cronológico) que este leyó.
3. Para un mensaje particular, los íconos deben mostrarse en orden de los IDs usuarios que leyeron el mensaje.
4. Un usuario no debe nunca visualizar su propio ícono debajo de ningún mensaje.

Conoces el tiempo en el que cada mensaje fue enviado y los tiempos de acceso de los usuarios. Puedes asumir que la comunicación es perfecta y cada mensaje es entregado instantáneamente a cada usuario. Dado un tiempo de acceso T_q de un usuario V_q, devuelve los IDs de los usuarios cuyos íconos son mostrados a ese usuario V_q debajo del último mensaje enviado hasta ese instante T_q.

Input

La primera línea contiene la cantidad de mensajes n ($1 \leq n \leq 2 \times 10^5$) que fueron enviados en el grupo y la cantidad de miembros v ($1 \leq v \leq 2 \times 10^5$) del grupo. La segunda línea contiene n números enteros ordenados y distintos tm_i ($1 \leq tm_i \leq 10^9$). El i-ésimo de ellos representa el tiempo en el cual el i-ésimo mensaje fue enviado. La tercera línea contiene la cantidad m ($1 \leq m \leq 2 \times 10^5$) de accesos a los mensajes del grupo. Cada una de las siguientes m líneas contiene dos números enteros v_i ($1 \leq v_i \leq v$) y t_i ($1 \leq t_i \leq 10^9$) representando que el i-ésimo acceso es del usuario v_i y ocurrió en el tiempo t_i. La última línea contiene dos números enteros V_q ($1 \leq V_q \leq v$) y T_q ($1 \leq T_q \leq 10^9$), donde V_q es el usuario para quien debemos mostrar los íconos de notificación y T_q es el tiempo del acceso. Nota que este acceso no necesariamente aparece en la lista de m accesos descrito arriba.

Output

En una sola línea, imprime K números enteros en orden ascendente, separados por un espacio, que corresponden a los IDs de los K usuarios cuyos íconos se les mostrarían al usuario V_q debajo del mensaje más cercano al tiempo T_q. Si no se muestra ningún ícono, imprime la frase "Nothing to display".

Example

standard input	standard output
5 3 1 3 5 7 11 7 1 2 1 3 1 4 2 3 2 5 3 1 3 2 1 4	2
5 3 1 3 5 7 11 7 1 5 1 6 1 7 2 1 2 2 2 3 3 11 2 1	Nothing to display
6 5 1 2 3 5 8 13 7 4 8 4 9 4 10 1 8 2 8 3 10 5 8 1 9	2 4 5

Problem F. Finding the Origins

Mientras comía uno de sus deliciosos minifaraones, el profesor Farnsworth empezó a preguntarse sobre su cultura y cómo acabaron teniendo una ciudad con un sistema de caminos dirigidos de tal forma que se pudiera llegar a todos los puntos de interés desde la Pirámide principal, y sólo existieran unos pocos caminos. De hecho, se sabe que al final de su civilización habían N puntos de interés (incluyendo la Pirámide principal) y exactamente $N-1$ caminos dirigidos. También se sabe que la configuración final de los caminos se obtuvo destruyendo algunos (tal vez ninguno) de los caminos originales, y que al principio no existían ciclos (ni siquiera autociclos) formados por los caminos. Pero el hecho más interesante de todos, es que la configuración final de la ciudad forma algo que ellos llaman un Árbol Dominante del original. Por alguna razón pensaron que ayudaría a optimizar algunas tareas. En un Árbol Dominante de una ciudad,

1. Un punto de interés A domina a un punto de interés B si cada camino desde la Pirámide principal a B debe pasar por A, y
2. Se dice que A domina inmediatamente a B si A es el único punto de interés que domina estrictamente a B pero no domina estrictamente a ningún otro punto de interés que domine estrictamente a B. Cada punto de interés, excepto la Pirámide principal, tiene un dominador inmediato. Cada punto de interés domina inmediatamente a lo más a 12 otros puntos (y esto es muy importante por razones de seguridad)

Construyendo un camino de A a B si y sólo si A domina inmediatamente a B, terminamos con el Árbol Dominante de la ciudad. Sabiendo todo esto, el profesor Farnsworth quiere que le digas cuántos orígenes distintos pudo tener la ciudad.

Input

Una línea que contiene N ($1 \leq N \leq 25$), el número de puntos de interés. El id de la Pirámide principal es 1. Luego siguen $N-1$ líneas, describiendo la configuración final de la ciudad. Cada línea con dos enteros A y B ($1 \leq A, B \leq N$), significa que hay un camino dirigido desde el punto de interés A al B.

Output

Una línea con un número, que es el resto de dividir el número de orígenes distintos que podría tener la ciudad entre 83921.

Example

standard input	standard output
4 1 2 1 3 3 4	5
4 1 2 1 3 1 4	25
6 1 2 1 3 1 4 1 5 1 6	29281

Problem G. Go Kill the Monsters

Estás jugando un nuevo juego. Existen n monstruos en este juego, cada uno con una fuerza específica. La fuerza del i-ésimo monstruo es a_i, lo que significa que cuesta a_i monedas de energía eliminar ese monstruo "a mano". En el juego hay, además, un **botón nuclear**, que puede ser presionado en varios momentos, tantas veces como desees. Presionarlo cuesta K monedas de energía; y consecuentemente, la mitad de los monstruos serán eliminados instantáneamente sin costo adicional. En resumen, si actualmente hay m monstruos vivos, luego de presionar el **botón nuclear**, $\lfloor \frac{m}{2} \rfloor$ monstruos al azar serán eliminados. Dados la fuerza de todos los monstruos y el costo de presionar el **botón nuclear**, encuentra el costo total mínimo (en monedas de energía) necesario para eliminar a todos los monstruos, en el peor caso.

Input

La primera línea contiene dos números separados por espacio, n, K ($1 \leq n \leq 10^5, 1 \leq K \leq 10^9$), que representan al número de monstruos y al costo de presionar el **botón nuclear**, respectivamente. La segunda línea contiene n enteros, $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), ellos son la fuerza de cada monstruo.

Output

Imprime un entero siendo el costo total mínimo necesario para eliminar a todos los monstruos en el peor caso.

Example

standard input	standard output
5 3 3 3 3 4 5	14

Problem H. Hurrying Back Home

El simpático robot WALL-E se encuentra colectando desechos en el planeta Tierra, el cual se puede representar como un plano cartesiano de dos dimensiones. Inicialmente, WALL-E se encuentra en la estación base con coordenada $(0, 0)$ en este plano, mirando hacia el norte, o sea, orientado hacia el eje-Y positivo. En sus tareas de búsqueda de desecho a colectar, WALL-E ejecuta una cantidad de operaciones de los siguientes tipos:

- L: Gira a la izquierda 90 grados
- R: Gira a la derecha 90 grados
- $F\ x$: Avanza x unidades de distancia en la dirección en la que está orientado

Al final de esos pasos, WALL-E encuentra un desecho y ahora debe volver a la estación base. Como su nivel de energía se encuentra bajo, WALL-E necesita regresar a la estación base en la mínima cantidad de operaciones posible. En particular, la operación del tipo "$F\ x$" cuenta como una sola operación sin importar la distancia x que WALL-E recorre en esa operación. Además, la orientación final con la que WALL-E llega a la coordenada $(0, 0)$ no importa.

Input

La primera línea de entrada contiene un número entero N ($1 \leq N \leq 100$) que representa la cantidad de operaciones que WALL-E ejecuta antes de encontrar un desecho. Las siguientes N líneas contienen, en orden cronológico, las operaciones que WALL-E ejecutó, en el formato descrito arriba. Para la operación del tipo "$F\ x$", x es un número entero en el rango de 1 a 100.

Output

Imprime un número entero que indica la menor cantidad de operaciones que WALL-E necesita para volver a la base en $(0, 0)$.

Example

standard input	standard output
1 F 10	3
7 F 10 L F 10 L F 10 L F 10	0

Problem I. Inspecting Trees

Se tiene un árbol con n vértices numerados de 1 a n. Cada nodo i tiene un valor inicial W_i. Se necesita ejecutar dos tipos de consultas:

1. *add(v, x)*: Incrementar el valor de todos los vecinos de v en x unidades. Se garantiza que $1 \leq v \leq n$ y $1 \leq x \leq 1000$.
2. *get(v)*: Imprimir el valor del vértice v ($1 \leq v \leq n$).

Input

La primera línea de la entrada contiene la cantidad de vértices n ($1 \leq n \leq 10^5$). La siguiente línea contiene n enteros W_i ($1 \leq W_i \leq 1000$) separados por un espacio en blanco con los valores iniciales de los vértices. Las siguientes $n-1$ líneas contienen dos enteros u y v ($1 \leq u, v \leq n$, $u \neq v$) separados por un espacio indicando que existe una arista entre los vértices u y v. La siguiente línea contiene el entero q ($1 \leq q \leq 2 \times 10^5$) que representa la cantidad de consultas a ejecutar. Las siguientes q líneas contienen la descripción de las consultas, en el formato descrito anteriormente.

Output

Para cada consulta de tipo 2, imprima el valor del vértice v en cada caso.

Example

standard input	standard output
4 1 2 3 4 1 2 1 3 1 4 5 1 1 3 2 1 2 2 2 3 2 4	1 5 6 7
4 1 2 3 4 1 2 1 3 1 4 6 1 2 3 2 1 2 2 2 3 1 1 4 2 4	4 2 3 8

Problem J. Join the Game

Hay tres pilas de piedras, cada una con A, B y C piedras. Puedes realizar la siguiente operación varias veces (posiblemente cero):

- Seleccionar dos pilas **no vacías**, quitar una piedra de cada pila seleccionada y agregar esas dos piedras a la tercera pila (la que no seleccionaste).

El objetivo es poner todas las piedras en una sola pila. ¿Puedes hacerlo?

Input

La primera línea contiene un número entero t ($1 \leq t \leq 10^4$) — el número de casos de prueba. A continuación, se describe cada caso de prueba. La primera línea de cada caso de prueba contiene tres enteros A, B y C ($0 \leq A, B, C \leq 10^4$) — el número de piedras en cada pila. Se garantiza que la suma de $A + B + C$ en todos los casos de prueba no excede $3 \cdot 10^4$.

Output

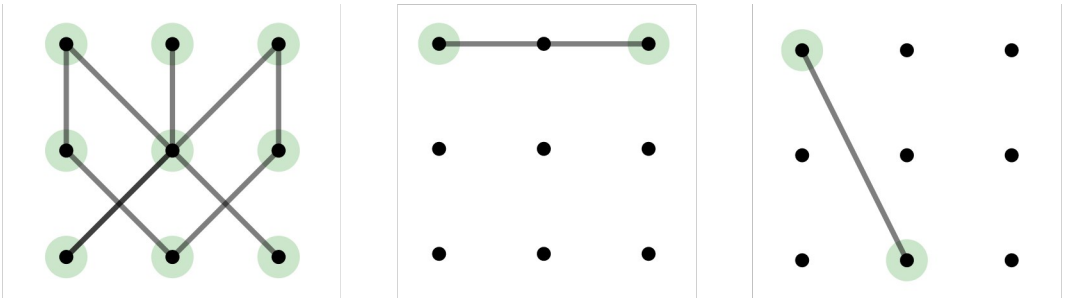
Para cada caso de prueba: Imprime en una sola línea "Yes" (sin comillas) si hay una forma de poner todas las piedras en una pila. De lo contrario, imprime "No" (sin comillas). Si hay una solución, en la siguiente línea imprime una cadena S ($|S| = m, 0 \leq m \leq 2 \cdot 10^4$). Cada caracter S_i ($1 \leq i \leq m$) de la cadena debe ser "A", "B" o "C", y denota el id de la tercera pila (la que no seleccionaste) en la i-ésima operación (si la cadena está vacía, imprime una línea vacía). Se garantiza que si existe una solución, entonces existe una construcción que utiliza a lo más $2 \cdot 10^4$ operaciones.

Example

standard input	standard output
5 9 8 2 0 0 0 20 30 40 5 5 5 9 22 14	Yes CCAAAAAA Yes No Yes BBBBB No

Problem K. Keyboard Patterns

Jack y Mary suelen competir en sus noches uno contra otro. Esta vez, Mary trajo un móvil nuevo para presumir de su bonito patrón de desbloqueo. Mary siempre está buscando una manera de competir con Jack. Esta vez, le preguntó a Jack "¿cuántos patrones cree que se pueden crear con mi teléfono móvil?" Jack miró detenidamente el teléfono y se dio cuenta de que había algunas teclas defectuosas que no se



Ejemplos de patrones en un teclado.

podían utilizar para crear un patrón. Mary explicó amablemente a Jack las limitaciones de los patrones que le interesaban:

- Un patrón puede comenzar en cualquier tecla no dañada y puede continuar moviéndose libremente hacia cualquier tecla.
- El usuario puede deslizar su dedo sobre una tecla defectuosa, pero no contribuirá al patrón actual.
- Para pasar a la siguiente tecla del patrón, el usuario DEBE deslizar el dedo en línea recta.
- Una tecla sólo puede utilizarse una vez para contribuir al patrón, es decir, una vez que el usuario desliza su dedo sobre la tecla la primera vez, esta tecla no contribuirá más al patrón actual aún si el usuario vuelve a deslizar el dedo sobre esta tecla.
- Desde su ubicación actual en la pantalla, el usuario **debe** deslizar su dedo en línea recta para llegar a la próximo tecla no defectuosa disponible. Ver la explicación del cuarto ejemplo abajo.
- El patrón termina cuando el usuario levanta su dedo de la pantalla o cuando ya no hay disponible teclas sin daño.
- Dos patrones P y Q, son diferentes si tienen diferentes tamaños o existe un índice i tal que $P_i \neq Q_i$.

Jack te ha pedido ayuda para no perder contra Mary en esta nueva competición. Dada la descripción del teléfono, debes calcular cuántos patrones se pueden crear utilizando las teclas no defectuosas del teclado. Considere el siguiente ejemplo para tener una comprensión más clara del problema. Supongamos que numeramos las teclas como un teclado numérico del 1 al 9 empezando por la esquina superior izquierda y terminando en la esquina inferior derecha. Si el usuario realiza el movimiento $1 \to 3$, el patrón generado es $1 \to 2 \to 3$ a menos que la tecla del número 2 sea defectuosa o se haya utilizado antes. En el cuarto ejemplo, si el usuario quiere pasar de la tecla 1 a la tecla 3, debe ir en línea recta pasando por la tecla 2, por lo que el patrón $1 \to 3 \to 2$ es imposible de lograr.

Input

La primera línea contiene un número entero $T \leq 1000$ que representa el número de teclados que se van a analizar. Las siguientes $3 \times T$ líneas contienen información sobre los teclados móviles. Cada teclado se representa como una matriz de 3×3 con los símbolos X y ., donde el símbolo X representa una tecla dañada y el símbolo . representa una tecla en buen estado.

Output

La salida contiene T líneas con el número de patrones que pueden crearse para cada teclado móvil dado en la entrada.

Example

standard input	standard output
6 XXX X.X XXX ..X XXX XXX .X. XXX X.X ... XXX XXX X.X .X. X.X ..X X.X ...	1 4 15 11 64 1168

Problem L. Luminaries

Se tienen N ($1 \leq N \leq 10^5$) puntos en el plano, representados por el par (x_i, y_i) para el i-ésimo punto. Cada punto puede estar solo en dos estados: activo e inactivo. Inicialmente, todos los puntos están activos. A estos, se les aplica una serie de operaciones, las cuales son descritas a continuación:

- R a
  Se rotan todos los puntos activos en a grados con respecto al origen $(0, 0)$, en sentido contrario a las manecillas del reloj. Con a entero y $a \in \{0, 90, -90, 180, -180, 270, -270\}$.
- T t_x t_y
  Se trasladan todos los puntos activos: sea (x_i, y_i) un punto activo, su nuevo valor pasa a ser $(x_i + t_x, y_i + t_y)$. Con t_x y t_y enteros y $-5 * 10^4 \leq t_x, t_y \leq 5 * 10^4$.
- M m_x m_y
  Se aplica la siguiente operación de escalado a todos los puntos activos: sea (x_i, y_i) un punto activo, su nuevo valor pasa a ser $(x * m_x, y * m_y)$. Con m_x y m_y enteros y $-5 * 10^4 \leq m_x, m_y \leq 5 * 10^4$.
- D d_x d_y
  Se aplica la siguiente operación de escalado a todos los puntos activos: sea (x_i, y_i) un punto activo, su nuevo valor pasa a ser $(x_i/d_x, y_i/d_y)$. Se asegura que x_i es divisible por d_x y que y_i es divisible por d_y. Con d_x y d_y enteros y $-5 * 10^4 \leq d_x, d_y \leq 5 * 10^4$.
- B i
  El i-ésimo punto pasa a inactivo si está activo, y a activo si está inactivo. Si un punto está inactivo, las operaciones antes descritas no le hacen efecto.
- P i
  Se debe imprimir las coordenadas del i-ésimo punto, independientemente de si está activo o no.

Input

- La primera línea contiene el entero N ($1 \leq N \leq 10^5$) la cantidad total de puntos y el entero Q ($1 \leq Q \leq 10^5$) la cantidad de operaciones.
- Las siguientes N líneas contienen dos enteros x_i, y_i ($-5 * 10^4 \leq x_i \leq 5 * 10^4$, $-5 * 10^4 \leq y_i \leq 5 * 10^4$) siendo estos los valores x y y del i-ésimo punto ($1 \leq i \leq N$).
- Las siguientes Q líneas contienen las operaciones a realizar sobre los puntos, donde al inicio se tiene el caracter $c_i \in R, T, B, M, D, P$ y separado por un espacio uno o dos enteros en correspondencia a la operación a realizar, ver sección anterior para más detalles.

Output

Por cada operación de impresión (las denotadas por el caracter P) en el mismo orden en que son solicitadas, se debe imprimir una línea con las coordenadas x y y del punto a operar.

Example

standard input	standard output
5 12 0 0 1 1 2 2 3 1 -2 -5 R 90 B 5 T 5 0 B 4 B 1 R -90 B 5 B 1 T -3 0 B 4 T -1 -1 P 1	1 -1
4 19 1 1 1 -1 -1 -1 -1 1 M 2 2 B 4 B 3 D 2 1 T 3 2 B 3 D 2 2 P 1 B 2 P 4 B 4 R 180 B 3 B 2 D 2 2 P 1 P 2 P 3 P 4	2 2 -2 2 -1 -1 1 0 1 1 1 -1

Problem M. Queries on Graphs

Tenemos un grafo G simple conectado no dirigido con n vértices numerados de 1 a n y m aristas. Cada nodo i tiene un valor inicial W_i. Se necesita ejecutar dos tipos de consultas:

1. *add(v, x)*: Incrementa el valor de todos los vecinos de v en $+x$ unidades, tal que $1 \leq x \leq 1000$.
2. *get(v)*: Imprimir el valor del vértice v ($1 \leq v \leq n$).

En un grafo *simple*,

1. Para cada pareja de nodos u y v, no existe más de una arista conectando u y v.
2. Para todos los vertices u, no existe una arista de u hacia sí mismo.

Input

La primera línea de la entrada contiene la cantidad de vértices n ($1 \leq n \leq 10^5$) y la cantidad de aristas m ($1 \leq m \leq 2 \times 10^5$) del grafo G. La siguiente línea contiene n enteros W_i ($1 \leq W_i \leq 1000$) separados por un espacio en blanco con los valores iniciales de los vértices. Las siguientes m líneas contienen dos enteros u y v ($1 \leq u, v \leq n$ y $u \neq v$) separados por un espacio que representan las aristas del grafo. La siguiente línea contiene el entero q ($1 \leq q \leq 2 \times 10^5$) que representa la cantidad de consultas a ejecutar. Las siguientes q líneas contienen la información de las consultas, como se explicó anteriormente.

Output

Para cada consulta de tipo 2, imprima el valor del vértice v en cada caso.

Example

standard input	standard output
5 5 1 2 3 4 5 1 2 1 3 1 4 2 4 5 2 6 1 1 3 1 5 1 2 1 2 2 2 3 2 4	1 6 6 7
5 6 1 2 3 4 5 1 2 1 3 1 4 2 4 5 2 5 3 6 1 1 1 2 1 2 2 1 2 3 2 3 2 4	1 3 4 8