Horses

Mansur ama criar caballos , al igual que sus antepasados. Ahora tiene la manada más grande de Kazajistán. Pero esto no siempre fue así. Hace N años, Mansur era sólo un Dzhigit (Termino en Kazajo para *un jóven*) y él sólo tenía un caballo. Soñaba con hacer un montón de dinero y finalmente llegar a ser un Bai (Termino en Kazajo para *una persona muy rica*).

Siendo el número de los años de 0 a $N-1$ en orden cronológico (es decir, el año $N-1$ es el más reciente) . El clima de cada año influyó en el crecimiento de la manada. Para cada año i, Mansur recuerda un número entero positivo que representa el coeficiente de crecimiento $X[i]$. Si ha iniciado el año i con h caballos, terminó el año con $h \cdot X[i]$ caballos en su manada.

Los caballos sólo podían ser vendidos al final de un año. Para cada año i, Mansur recuerda un entero positivo $Y[i]$: el precio por el que podría vender un caballo al final del año i. Después de cada año, era posible vender arbitrariamente muchos caballos, cada uno al mismo precio $Y[i]$.

Mansur se pregunta: ¿cuál es la mayor cantidad de dinero que podría tener ahora si hubiese elegido los mejores momentos para vender sus caballos durante los N años?. Usted tiene el honor de ser un invitado en la casa de Mansur en sus toi (Termino en Kazajo para *vacaciones*), y él le pidió que responda a esta pregunta.

La memoria de Mansur mejora durante toda la noche, así que hace una secuencia de M actualizaciones. Cada actualización cambiará ya sea uno de los valores de $X[i]$ o uno de los valores $Y[i]$. Después de cada nueva actualización le pide la mayor cantidad de dinero que podría haber ganado con la venta de sus caballos. Las actualizaciones de Mansur son acumulativas: cada una de sus respuestas debe tener en cuenta todas las actualizaciones anteriores. Tenga en cuenta que un $X[i]$ o $Y[i]$ puede ser actualizado varias veces.

Las respuestas reales a las preguntas de Mansur pueden ser enormes. Con el fin de evitar trabajar con un grandes números, sólo está obligado a reportar las respuestas modulo $10^9 + 7$

Example

Supongamos que son $N = 3$ años, con la siguiente información:

	0	**1**	**2**
X	2	1	3
Y	3	4	1

Para estos valores iniciales, Mansur puede ganar más si vende sus dos caballos al final del año 1. Todo el proceso se verá de la siguiente manera:

- Inicialmente, Mansur tiene 1 caballo.

- Despues del año 0 podía tener $1 \cdot X[0] = 2$ caballos.
- Despues del año 1 podía tener $2 \cdot X[1] = 2$ caballos.
- El ahora puede vender sus dos caballos. El total de ganancia será $2 \cdot Y[1] = 8$.

Entonces, supongamos que $M = 1$ es actualizado: cambiando $Y[1]$ a 2.

Después de actualizar tenemos:

	0	1	2
X	2	1	3
Y	3	2	1

En este caso, una de las soluciones óptimas es vender un caballo despues del año 0 y tres caballos despues del año 2.

Todo el proceso se verá de la siguiente manera :

- Inicialmente, Mansur tiene 1 caballo.
- Despues del año 0 tendrá $1 \cdot X[0] = 2$ caballos.
- Ahora puede vender uno de esos caballos por $Y[0] = 3$, y le queda un caballo.
- Despues del año 1 podrá tener $1 \cdot X[1] = 1$ caballos.
- Despues del año 2 podrá tener $1 \cdot X[2] = 3$ caballos.
- Ahora puede vender estos tres caballos por$3 \cdot Y[2] = 3$. La cantidad total de dinero es $3 + 3 = 6$.

Task

Se le da N, X , Y , y la lista de cambios . Antes de la primera actualización , y después de cada actualización, calcule la cantidad máxima de dinero que Mansur podría conseguir por sus caballos, módulo $10^9 + 7$. Usted debe implementar las funciones `init`, `updateX` y `updateY`.

- `init(N, X, Y)` — El grader llamará esta función primero y sólo una vez.
  - `N`: el número de años.
  - `X`: un arreglo de tamaño N. Para $0 \leq i \leq N - 1$, $X[i]$ es el coeficiente de crecimiento para el año i.
  - `Y`: un arreglo de tamaño N. Para $0 \leq i \leq N - 1$, $Y[i]$ es el precio de un caballo para el año i.
  - Observe que ambos `X` y `Y` especifican los valores iniciales dados por Mansur (antes de cualquier actualizacion)
  - Luego de terminar la función `init` los arreglos X y Y permanecen válidos, y usted puede modificar su contenido si lo considera necesario.
  - La función debe devolver la cantidad máxima de dinero que Mansur podría conseguir para

estos valores iniciales de X and Y, módulo $10^9 + 7$.

- `updateX(pos, val)`
  - `pos`: un entero en el rango $0, \ldots, N-1$.
  - `val`: el nuevo valor para $X[pos]$.
  - La función debe devolver la cantidad máxima de dinero Mansur podría conseguir después de esta actualización, módulo $10^9 + 7$.
- `updateY(pos, val)`
  - `pos`: un entero en el rango $0, \ldots, N-1$.
  - `val`: el nuevo valor para $Y[pos]$.
  - La función debe devolver la cantidad máxima de dinero que Mansur podría conseguir después de esta actualización, módulo $10^9 + 7$.

Puede asumir que todos los valores iniciales, así como cada actualizacion de $X[i]$ y $Y[i]$ son entre 1 y 10^9 inclusive.
Despues de ejecutar `init`, el grader ejecutara `updateX` y `updateY` muchas veces. El número total de llamadas a `updateX` y `updateY` será M.

Subtasks

subtask	puntos	N	M	restricciones adicionales
1	17	$1 \le N \le 10$	$M = 0$	$X[i], Y[i] \le 10$, $X[0] \cdot X[1] \cdot \ldots \cdot X[N-1] \le 1,000$
2	17	$1 \le N \le 1,000$	$0 \le M \le 1,000$	ninguna
3	20	$1 \le N \le 500,000$	$0 \le M \le 100,000$	$X[i] \ge 2$ y $val \ge 2$ para `init` y `updateX` correspondiente
4	23	$1 \le N \le 500,000$	$0 \le M \le 10,000$	ninguna
5	23	$1 \le N \le 500,000$	$0 \le M \le 100,000$	ninguna

Implementation details

Debes enviar exactamente un archivo, llamado `horses.c`, `horses.cpp`, `horses.pas`, o `horses.java`.
Este archivo debe implementar los subprogramas descritos anteriormente como funciones o métodos , utilizando las siguientes especificaciones.

programas en C/C++ (incluye `horses.h` en el tope del codigo fuente)

```
int init(int N, int X[], int Y[]);
int updateX(int pos, int val);
int updateY(int pos, int val);
```

programas en Pascal (aplicar el método descrito en la unidad `horses`)

```
function init(N : longint; var X, Y : array of longint) : longint;
function updateX(pos, val : longint) : longint;
function updateY(pos, val : longint) : longint;
```

programas en Java (aplicar el método descrito en la clase pública `horses`)

```
public int init(int N, int[] X, int[] Y);
public int updateX(int pos, int val);
public int updateY(int pos, int val);
```

Sample grader

El grader ejemplo lee la entrada desde el archivo `horses.in` en el siguiente formato:

- línea 1: `N`
- línea 2: `X[0]` … `X[N - 1]`
- línea 3: `Y[0]` … `Y[N - 1]`
- línea 4: `M`
- líneas 5, …, `M` + 4: tres números `type pos val` (`type=1` para `updateX` y `type=2` para `updateY`).

El grader ejemplo imprime el valor de retorno de `init` y luego los valores retornados de todas las llamadas a `updateX` y `updateY`.