| TASK | MORTADELA | KRIZALJKA | LANCI | POPUST | INFORMACIJE | INSPEKTOR |
|---|---|---|---|---|---|---|
| **source code** | `mortadela.pas`<br>`mortadela.c`<br>`mortadela.cpp` | `krizaljka.pas`<br>`krizaljka.c`<br>`krizaljka.cpp` | `lanci.pas`<br>`lanci.c`<br>`lanci.cpp` | `popust.pas`<br>`popust.c`<br>`popust.cpp` | `informacije.pas`<br>`informacije.c`<br>`informacije.cpp` | `inspektor.pas`<br>`inspektor.c`<br>`inspektor.cpp` |
| **input** | standard input (*stdin*) | | | | | |
| **output** | standard output (*stdout*) | | | | | |
| **time limit** | 1 second | 1 second | 1 second | 2 seconds | 1 second | 4 seconds |
| **memory limit** | 32 MB | 32 MB | 32 MB | 32 MB | 32 MB | 32 MB |
| **point value** | **50** | **80** | **100** | **120** | **140** | **160** |
| | **650** | | | | | |

Problems translated from Croatian by: **Ivan Pilat**

The National Supermarket Chain (NSC) likes to boast that it has the lowest price for mortadella in the country. In fact, if a customer manages to find cheaper mortadella in any other chain, the NSC will match the price for that customer.

Matej and Filip decided to accept that challenge. They will visit **N different supermarket chains** in order to find mortadella not only cheaper than the one in NSC, but the **cheapest on the market**. If they are successful, they will be able to buy the cheapest mortadella in an NSC branch close to their school.

NSC was hoping that no one would be able to find cheaper mortadella since all supermarket chains (including NSC) express mortadella prices in a convoluted way: **X dollars for Y grams of mortadella**.

Write a program to, given mortadella prices in NSC as well as the remaining **N** chains, determine the price that Matej and Filip will have to pay for **1000 grams of mortadella** in the NSC close to their school.

## INPUT

The first line of input contains two positive integers $\mathbf{X_{NSC}}$ ($1 \leq \mathbf{X_{NSC}} \leq 100$) and $\mathbf{Y_{NSC}}$ ($1 \leq \mathbf{Y_{NSC}} \leq 1000$), where $\mathbf{X_{NSC}}$ is the price of $\mathbf{Y_{NSC}}$ grams of mortadella in the NSC chain.

The second line of input contains the positive integer **N** ($1 \leq \mathbf{N} \leq 100$), the number of supermarket chains (excluding NSC).

Each of the following **N** lines contains two positive integers $\mathbf{X_i}$ ($1 \leq \mathbf{X_i} \leq 100$) and $\mathbf{Y_i}$ ($1 \leq \mathbf{Y_i} \leq 1000$), **i**=1..**N**, where $\mathbf{X_i}$ is the price of $\mathbf{Y_i}$ grams of mortadella in the $\mathbf{i^{th}}$ supermarket chain.

## OUTPUT

The first and only line of output must contain the requested real number (price). It is allowed to differ at most 0.01 from the exact solution.

## SAMPLE TESTS

| **input** | **input** | **input** |
|---|---|---|
| 5 100<br>3<br>4 100<br>3 100<br>7 100 | 13 6<br>5<br>56 679<br>35 120<br>99 999<br>56 73<br>37 532 | 100 5<br>3<br>99 8<br>65 14<br>78 10 |
| **output** | **output** | **output** |
| 30.00 | 69.55 | 4642.86 |

Do you like to solve crosswords? If you do, you probably know the traditional lead-in for beginners – small 3 by 3 crosswords, like the following[1]:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | | |
| 3 | | |

Horizontal:
1. device used to cool a PC
2. solid water
3. to obtain

Vertical:
1. small, soft, sweet fruit
2. strong playing card
3. fisherman's tool

Solve a crossword like this one! Not gonna happen? Fine, then at least put it together given six words that appear in it (**three as horizontal** and **three as vertical**).

## INPUT

Each of the six lines of input contains a word consisting of three uppercase English letters. The words are given in a lexicographically sorted order.

## OUTPUT

If it is impossible to put together a 3 by 3 crossword using the given six words, output 0.

Otherwise, output the crossword in three lines.

If there is more than one possible solution, output the one that is first in lexicographic order, where, for sorting purposes, the crossword is viewed as a nine-letter word obtained by concatenating its rows.

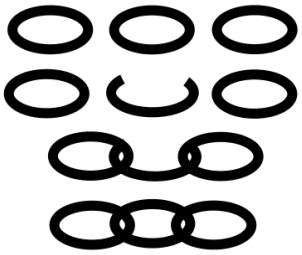
## SAMPLE TESTS

| **input** | **input** | **input** |
|---|---|---|
| ANA<br>ANA<br>DAR<br>DAR<br>RAD<br>RAD | EVO<br>HEP<br>HIR<br>IVA<br>PAD<br>ROD | AKO<br>CES<br>DOC<br>DON<br>ESI<br>KES |
| **output** | **output** | **output** |
| DAR<br>ANA<br>RAD | HEP<br>IVA<br>ROD | 0 |

[1] adapted from http://www.goobix.com/crosswords/0303/

Mirko has found **N** chains in his attic. Each chain consists of some number of links, where each link has at most two adjacent links. Each link can be opened or closed, so it is possible to separate chains or connect them into a longer chain. Mirko would like to connect all chains into one huge chain, while **opening and closing as few links as possible**.

For example, if Mirko has only three chains, each consisting of only one link, he can open one of them, use it to connect the remaining two and close it:



Given the number of chains and the length of each chain, find the minimum number of links that Mirko has to open and close in order to bind them all in ~~darkness~~ one long chain.

## INPUT

The first line of input contains the positive integer **N** ($2 \leq \mathbf{N} \leq 500\,000$), the number of chains.

The second line of input contains **N** positive integers $\mathbf{L_i}$ ($1 \leq \mathbf{L_i} \leq 1\,000\,000$), the lengths of individual chains.

## OUTPUT

The first and only line of output must contain the required minimum number of links.

## SAMPLE TESTS

| **input** | **input** | **input** |
|---|---|---|
| 2<br>3 3 | 3<br>1 1 1 | 5<br>4 3 5 7 9 |
| **output** | **output** | **output** |
| 1 | 1 | 3 |

**Clarification of the first example:** Mirko can open the last link of the first chain, connect it with the first link of the second chain and close it.

**Clarification of the third example:** Here it is best to completely take apart the chain of length 3, using its three links to connect the remaining chains.

Mirko is hungry as a bear, scratch that, programmer and has stumbled upon a local restaurant. The restaurant offers **N** meals and has an interesting pricing policy: each meal **i** has two assigned prices, $\mathbf{A}_i$ and $\mathbf{B}_i$. Mirko pays **A** only for the **first ordered meal**, while **B** prices apply for **all other** meals.

Mikro can't decide how many meals to order. In order to make his decision easier, he has asked you to compute, for each **k** between 1 i **N** (inclusive), the minimum total price for **k** ordered meals. Mirko doesn't care which particular meals he orders or in which order he orders them, however he won't order the same meal twice. Order, order, order.

## INPUT

The first line of input contains the positive integer **N** ($2 \leq \mathbf{N} \leq 500\ 000$), the number of different meals offered by the restaurant.

Each of the following **N** lines contains two positive integers, $\mathbf{A}_i$ and $\mathbf{B}_i$ ($1 \leq \mathbf{A}_i, \mathbf{B}_i \leq 1\ 000\ 000\ 000$), the prices for meal **i** as described above.

## OUTPUT

Output must consist of **N** lines, where line **k** contains the minimum price for ordering exactly **k** different meals.

## SAMPLE TESTS

| **input** | **input** | **input** |
|---|---|---|
| 3<br>10 5<br>9 3<br>10 5 | 2<br>100 1<br>1 100 | 5<br>1000000000 1000000000<br>1000000000 1000000000<br>1000000000 1000000000<br>1000000000 1000000000<br>1000000000 1000000000 |
| **output** | **output** | **output** |
| 9<br>13<br>18 | 1<br>2 | 1000000000<br>2000000000<br>3000000000<br>4000000000<br>5000000000 |

**Clarification of the first example:**
k = 1: Mirko pays $A_2 = 9$ for the starting meal 2.
k = 2: Mirko pays $A_1 = 10$ for the starting meal 1, then $B_2 = 3$ for meal 2.
k = 3: Mirko pays $A_1 = 10$ for the starting meal 1, then $B_2 = 3$ for meal 2, and finally $B_3 = 5$ for meal 3.

Mirko was bored, so he took a piece of paper and wrote down a sequence **A** of length **N**, which contains each positive integer between **1** and **N**, inclusive, **exactly once**. After that, he took another piece of paper and wrote down **M** descriptions of the sequence **A**.

Each description has one of the following formats:

**1 x y v –** the largest number in positions between **x** and **y** (inclusive) equals **v**

**2 x y v –** the smallest number in positions between **x** and **y** (inclusive) equals **v**

Then Slavko came, saw, and stole the first paper. Mirko is desperate and has asked you to find some sequence matching the descriptions, not necessarily equal to the original sequence.

## INPUT

The first line of input contains two positive integers, **N** ($1 \leq \mathbf{N} \leq 200$), the length of the sequence, and **M** ($0 \leq \mathbf{M} \leq 40\,000$), the number of descriptions.

Each of the following **M** lines contains a description as described above.

## OUTPUT

The first and only line of output must contain a sequence of **N** space-separated positive integers (matching the descriptions and containing all positive integers from **1** to **N**), or -1 if no such sequence exists.

## SAMPLE TESTS

| **input** | **input** | **input** |
|---|---|---|
| 3 2<br>1 1 1 1<br>2 2 2 2 | 4 2<br>1 1 1 1<br>2 3 4 1 | 5 2<br>1 2 3 3<br>2 4 5 4 |
| **output** | **output** | **output** |
| 1 2 3 | -1 | 1 2 3 4 5 |

A new town was just inaugurated in a small country. As usual, Mirko has secured the position of the chief tax inspector. His duty is ensuring adequate accounting in all the different companies in the town.

There are **N** business offices along the main street, numbered 1 to **N** from left to right. All offices are empty in the beginning; in time, companies will move in and out of these offices. From time to time, Mirko will pass by some of the offices and inspect the accounting of only one company, the currently wealthiest one in those offices.

A company moving in is described by four integers:

- **T** – the move-in day, numbered from town inauguration (which is day 1),
- **K** – the office number,
- **Z** – the daily profit of the company (can be negative if the company is losing money),
- **S** – balance of the company's account on move-in day.

If there is already a company in office **K**, that company **moves out** when the new company moves in. The new company doesn't open for business (or earn profit) until the day after move-in.

Mirko's inspection stroll is described by three integers:

- **T** – the inspection day, numbered from town inauguration,
- **A** and **B** – Mirko will pass by all offices with numbers between **A** and **B**, inclusive.

Since Mirko works only at the end of the day, all companies will have already finished business and posted profit for the day by the time of Mirko's stroll.

Help Mirko by writing a program to find, for each stroll, the account balance of the currently wealthiest company that Mirko is passing by.

## INPUT

The first line of input contains two positive integers, **N** ($1 \leq \mathbf{N} \leq 100\,000$) and **M** ($1 \leq \mathbf{M} \leq 300\,000$), the number of offices and events, respectively.

Each of the following **M** lines contains a description of one event, formatted either as "1 **T K Z S**" (for company move-in) or as "2 **T A B**" (for Mirko's inspection stroll).

All events are given chronologically, and at most one event will happen each day (that is, **T** will be strictly increasing). The last event's day number will be less than $10^6$, and all **Z** and **S** numbers' absolute values will be less than $10^6$.

## OUTPUT

For each Mirko's stroll output a line containing the account balance of the company that Mirko will inspect, or the word "nema" (without quotes) if all offices that he will pass by are empty.

## SAMPLE TESTS

**input**

```
2 4
1 1 1 2 4
1 2 2 3 2
2 5 1 2
2 7 1 2
```

**output**

```
12
17
```

**input**

```
3 6
1 1 1 4 -2
1 2 2 2 6
2 3 3 1
2 4 3 1
1 5 3 -6 20
2 6 2 3
```

**output**

```
8
10
14
```

**input**

```
5 9
1 1 5 4 -5
2 2 3 5
1 3 4 6 9
2 4 1 2
1 6 2 2 3
2 8 2 1
1 9 4 0 17
2 10 5 5
2 11 1 4
```

**output**

```
-1
nema
7
31
17
```