Task: GAR
Garden

Official English Version

Nowy Sącz, POLAND 2005
The 17-th International
Olympiad in Informatics

Day 1. Source file `gar.*` *Saturday, 20–08–2005*

Available memory: 32 MB. Maximum running time: 0.5 s.

Byteman owns the most beautiful garden in Bytetown. He planted n roses in his garden. Summer has come and the flowers have grown big and beautiful. Byteman has realized that he is not able to take care of all the roses on his own. He has decided to employ two gardeners to help him. He wants to select two rectangular areas, so that each of the gardeners will take care of the roses inside one area. The areas should be disjoint and each should contain exactly k roses.

Byteman wants to make a fence surrounding the rectangular areas, but he is short of money, so he wants to use as little fence as possible. Your task is to help Byteman select the two rectangular areas.

The garden forms a rectangle l meters long and w meters wide. It is divided into $l \cdot w$ squares of size 1 meter $\times$ 1 meter each. We fix a coordinate system with axes parallel to the sides of the garden. All squares have integer coordinates (x,y) satisfying $1 \leq x \leq l$, $1 \leq y \leq w$. Each square may contain any number of roses.

The rectangular areas, which must be selected, should have their sides parallel to the sides of the garden and the squares in their corners should have integer coordinates. For $1 \leq l_1 \leq l_2 \leq l$ and $1 \leq w_1 \leq w_2 \leq w$, a rectangular area with corners (l_1, w_1), (l_1, w_2), (l_2, w_1) and (l_2, w_2):

- contains all the squares with coordinates (x,y) satisfying $l_1 \leq x \leq l_2$ and $w_1 \leq y \leq w_2$, and
- has perimeter $2 \cdot (l_2 - l_1 + 1) + 2 \cdot (w_2 - w_1 + 1)$.

The two rectangular areas must be disjoint, that is they cannot contain a common square. Even if they have a common side, or part of it, they must be surrounded by separate fences.

Task

Write a program, that:

- reads from the standard input the dimensions of the garden, the number of roses in the garden, the number of roses that should be in each of the rectangular areas, and the positions of the roses,
- finds the corners of two such rectangular areas with minimum sum of perimeters that satisfy the given conditions,
- writes to the standard output the minimum sum of perimeters of two non-overlapping rectangular areas, each containing exactly the given number of roses (or a single word `NO`, if no such pair of areas exists).

Input

The first line of standard input contains two integers: l and w ($1 \leq l, w \leq 250$) separated by a single space — the length and the width of the garden. The second line contains two integers: n and k ($2 \leq n \leq 5000$, $1 \leq k \leq n/2$) separated by a single space — the number of roses in the garden and the number of roses that should be in each of the rectangular areas. The following n lines contain the coordinates of the roses, one rose per line. The $(i+2)$-nd line contains two integers l_i, w_i ($1 \leq l_i \leq l$, $1 \leq w_i \leq w$) separated by a single space — the coordinates of the square containing the i-th rose. Two or more roses can occur in the same square.

In 50% of test cases, the dimensions of the garden will satisfy $l, w \leq 40$.

Output

The standard output should contain only one line with exactly one integer — the minimum sum of perimeters of two non-overlapping rectangular areas, each containing exactly *k* roses, or a single word `NO`, if no such pair of areas exists.

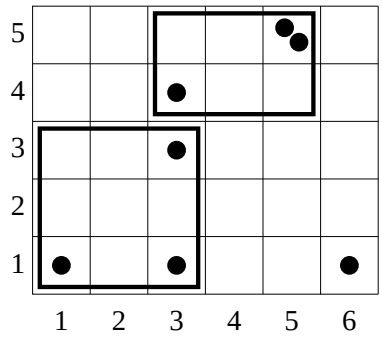
Example

For the input data:

```
6 5
7 3
3 4
3 3
6 1
1 1
5 5
5 5
3 1
```

the correct result is:

```
22
```



Task: MEA

Official English Version

Mean Sequence

Day 1. Source file `mea.*` *Saturday, 20–08–2005*

Available memory: 16 MB. Maximum running time: 5 s.

Consider a nondecreasing sequence of integers $s_1, \ldots, s_{n+1}$ ($s_i \leq s_{i+1}$ for $1 \leq i \leq n$). The sequence $m_1, \ldots, m_n$ defined by $m_i = \frac{1}{2}(s_i + s_{i+1})$, for $1 \leq i \leq n$, is called the *mean sequence* of sequence $s_1, \ldots, s_{n+1}$. For example, the mean sequence of sequence 1, 2, 2, 4 is the sequence 1.5, 2, 3. Note that elements of the mean sequence can be fractions. However, this task deals with mean sequences whose elements are integers only.

Given a nondecreasing sequence of n integers $m_1, \ldots, m_n$, compute the number of nondecreasing sequences of $n+1$ integers $s_1, \ldots, s_{n+1}$ that have the given sequence $m_1, \ldots, m_n$ as mean sequence.

Task

Write a program that:

- reads from the standard input a nondecreasing sequence of integers,
- calculates the number of nondecreasing sequences, for which the given sequence is mean sequence,
- writes the answer to the standard output.

Input

The first line of the standard input contains one integer n ($2 \leq n \leq 5\,000\,000$). The remaining n lines contain the sequence $m_1, \ldots, m_n$. Line $i+1$ contains a single integer m_i ($0 \leq m_i \leq 1\,000\,000\,000$). You can assume that in 50% of the test cases $n \leq 1\,000$ and $0 \leq m_i \leq 20\,000$.

Output

Your program should write to the standard output exactly one integer — the number of nondecreasing integer sequences, that have the input sequence as the mean sequence.

Example

For the input data:	the correct result is:
3 2 5 9	4

Indeed, there are four nondecreasing integer sequences for which $2, 5, 9$ is the mean sequence. These sequences are:

- $2, 2, 8, 10$,
- $1, 3, 7, 11$,
- $0, 4, 6, 12$,
- $-1, 5, 5, 13$.

Task: MOU
Mountain

Official English Version

Day 1. Source file `mou.*` *Saturday, 20–08–2005*

Available memory: 256 MB. Maximum running time: 3 s.

The Mountain Amusement Park has opened a brand-new simulated roller coaster. The simulated track consists of n rails attached end-to-end with the beginning of the first rail fixed at elevation 0. Byteman, the operator, can reconfigure the track at will by adjusting the elevation change over a number of consecutive rails. The elevation change over other rails is not affected. Each time rails are adjusted, the following track is raised or lowered as necessary to connect the track while maintaining the start at elevation 0. The figure on the next page illustrates two example track reconfigurations.

Each ride is initiated by launching the car with sufficient energy to reach height h. That is, the car will continue to travel as long as the elevation of the track does not exceed h, and as long as the end of the track is not reached.

Given the record for all the day's rides and track configuration changes, compute for each ride the number of rails traversed by the car before it stops.

Internally, the simulator represents the track as a sequence of n elevation changes, one for each rail. The i-th number d_i represents the elevation change (in centimetres) over the i-th rail. Suppose that after traversing $i-1$ rails the car has reached an elevation of h centimetres. After traversing i rails the car will have reached an elevation of $h+d_i$ centimetres.

Initially the rails are horizontal; that is, $d_i = 0$ for all i. Rides and reconfigurations are interleaved throughout the day. Each reconfiguration is specified by three numbers: a, b and D. The segment to be adjusted consists of rails a through b (inclusive). The elevation change over each rail in the segment is set to D. That is, $d_i = D$ for all $a \leq i \leq b$.

Each ride is specified by one number h — the maximum height that the car can reach.

Task

Write a program that:

- reads from the standard input a sequence of interleaved reconfigurations and rides,
- for each ride computes the number of rails traversed by the car,
- writes the results to the standard output.

Input

The first line of input contains one positive integer n — the number of rails, $1 \leq n \leq 1\,000\,000\,000$. The following lines contain reconfigurations interleaved with rides, followed by an end marker. Each line contains one of:

- Reconfiguration — a single letter '`I`', and integers a, b and D, all separated by single spaces ($1 \leq a \leq b \leq n$, $-1\,000\,000\,000 \leq D \leq 1\,000\,000\,000$).
- Ride — a single letter '`Q`', and an integer h ($0 \leq h \leq 1\,000\,000\,000$) separated by a single space;

- A single letter ‘E’ — the end marker, indicating the end of the input data.

You may assume that at any moment the elevation of any point in the track is in the interval $[0, 1\,000\,000\,000]$ centimetres. The input contains no more than 100 000 lines.

In 50 % of test cases n satisfies $1 \leq n \leq 20\,000$ and there are no more than 1 000 lines of input.

Output

The i-th line of output should consist of one integer — the number of rails traversed by the car during the i-th ride.

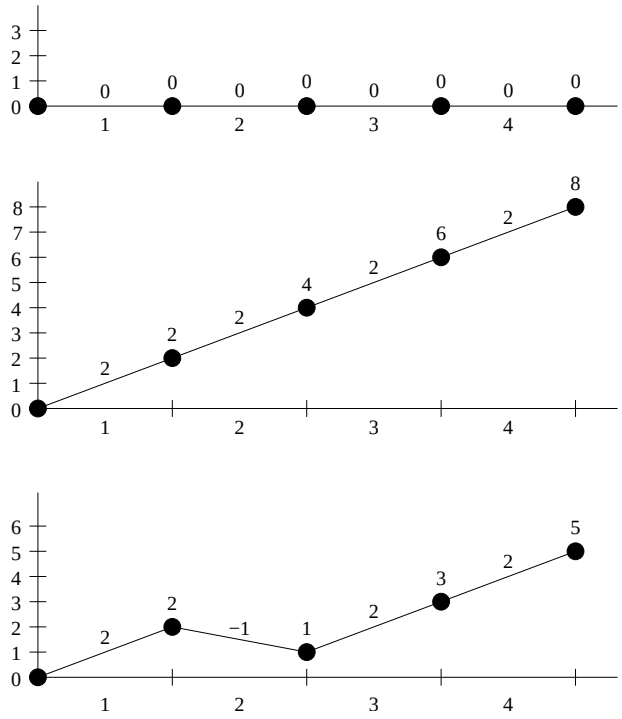
Example

For the input data:

```
4
Q 1
I 1 4 2
Q 3
Q 1
I 2 2 -1
Q 3
E
```

the correct result is:

```
4
1
0
3
```



Views of the track before and after each reconfiguration. The x axis denotes the rail number. The y axis and the numbers over points denote elevation. The numbers over segments denote elevation changes.