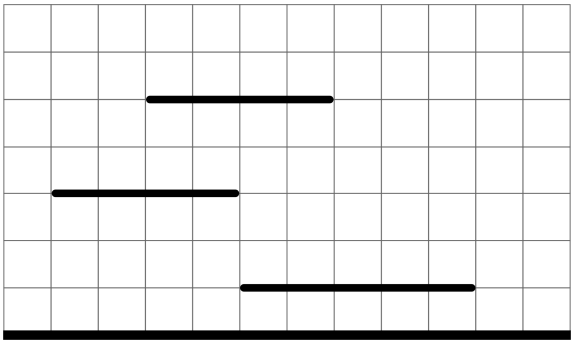
| TASK | PLATFORME | NIKOLA | KUHAR | JEDNAKOST |
|---|---|---|---|---|
| **input** | standard input (keyboard) | | | |
| **output** | standard output (screen) | | | |
| **time limit** (per test case) | 1 second | | | |
| **memory limit** (heap+stack) | 32 MB | | | 64 MB |
| **points** | **50** | **70** | **80** | **100** |
| | **300** | | | |

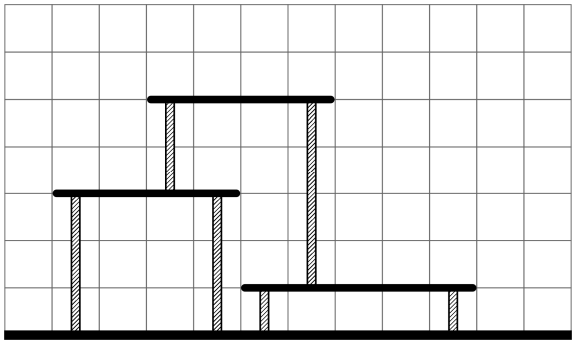
A level is being designed for a new platform game. The locations of the platforms have been chosen. Contrary to popular opinion, platforms can't float in the air, but need pillars for support. More precisely, each of the two ends of the platform needs to be supported by a **pillar standing on the floor or on a different platform**.

You will be given the locations of the platforms in a coordinate system as in the left image below. Each platform's location is determined by its altitude (vertical distance from the ground) and the start and end coordinates in the horizontal direction. Each support pillar is placed **half a unit** from the end of a platform, as in the right image.

Determine the total length of pillars needed to support all the platforms.



Example level with three platforms. The lowest platform is at altitude 1, the second lowest at altitude 3 and the third at altitude 5.



The total length of pillars needed to support all platforms is 14.

## Input

The first line contains the integer N, $1 \leq N \leq 100$, the number of platforms.

Each of the following N lines contains the position of one platform, three coordinates Y, $X_1$ and $X_2$. The first number is the altitude, the other two the horizontal coordinates. All coordinates will be positive integers less than 10000 satisfying $X_2 > X_1+1$ (i.e. the length of each platform will be at least 2).

The input will be such that no two platforms overlap.

## Output

Output the total length of pillars needed to support all the platforms.

## Examples

**input**

```
3
1 5 10
3 1 5
5 3 7
```

**output**

```
14
```

**input**

```
5
50 50 90
40 40 80
30 30 70
20 20 60
10 10 50
```

**output**

```
200
```

Against his own will, Nikola has become the main character in a game. The game is played on a row of N squares, numbered 1 to N. Nikola is initially in square 1 and can jump to other squares. Nikola's first jump must be to square 2. Each subsequent jump must satisfy two constraints:

- If the jump is in the forward direction, it must be one square longer than the preceding jump.
- If the jump is in the backward direction, it must be exactly the same length as the preceding jump.

For example, after the first jump (when on square 2), Nikola can jump back to square 1 or forwards to square 4.

Every time he enters a square, Nikola must pay an **entry fee**. Nikola's goal is to get from square 1 to square N **as cheaply as possible**. Write a program that determines the smallest total cost for Nikola to get to square N.

## Input

The first line contains the integer N, $2 \leq N \leq 1000$, the number of squares.

Each of the following N lines contains the entry fee for one square, a positive integer less than 500. The entry fees will be given in order for squares 1 to N.

## Output

Output the smallest total cost for Nikola to get to square N.

## Examples

**input**

```
6
1
2
3
4
5
6
```

**output**

```
12
```

**input**

```
8
2
3
4
3
1
6
1
4
```

**output**

```
14
```

In the first example, after jumping to square 2, Nikola jumps back to square 1. From there he can jump to square 3 and then to 6.

Lisa works as a waitress in a restaurant. Tonight is her birthday so Lisa asked the chef to prepare his special meal for her friends. The chef's meal is made of N ingredients. To prepare one serving of the meal he needs a certain amount of each ingredient.

There are some ingredients already available in the kitchen and Lisa will buy the rest at the grocery store. The store has all the necessary ingredients, each coming in **smaller** and **larger** packages. Lisa has M dollars and wants to spend them so that the chef can make **the most servings** of his meal.

## Input

The first line contains two integers N and M, $1 \leq N \leq 100$, $1 \leq M \leq 100000$.

Each of the following N lines contains 6 positive integers, information about one ingredient. These specify, in order:

- **X**, $10 \leq X \leq 100$, the amount of the ingredient needed in one serving;
- **Y**, $1 \leq Y \leq 100$, the amount of the ingredient already available in the kitchen;
- $\mathbf{S_M}$, $1 \leq S_M < 100$, the size of the smaller package at the store;
- $\mathbf{P_M}$, $10 \leq P_M < 100$, the price of the smaller package;
- $\mathbf{S_V}$, $S_M < S_V \leq 100$, the size of the larger package; and
- $\mathbf{P_V}$, $P_M < P_V \leq 100$, the price of the larger package.

## Output

Output the largest number of servings the chef can make if Lisa spends her money wisely.

## Examples

**input**

```
2 100
10 8 10 10 13 11
12 20 6 10 17 24
```

**output**

```
5
```

**input**

```
3 65
10 5 7 10 13 14
10 5 8 11 14 15
10 5 9 12 15 16
```

**output**

```
2
```

In the first example, for 99 dollars Lisa will buy three smaller and one larger package of the first ingredient, as well as one smaller and two larger packages of the second ingredient ($3 \cdot 10 + 1 \cdot 11 + 1 \cdot 10 + 2 \cdot 24 = 99$).

The chef will then have 51 units ($8 + 3 \cdot 10 + 1 \cdot 13$) of the first ingredient and 60 units ($20 + 1 \cdot 6 + 2 \cdot 17$) of the second ingredient, enough for 5 servings.

While browsing a math book, Mirko found a strange equation of the form A=S. What makes the equation strange is that A and S are not the same. Mirko realized that the left side of the equation should have addition operations between some pairs of digits in A.

Write a program that inserts the **smallest number** of addition operations on the left side to make the equation correct. The numbers in the corrected equation may contain arbitrary amounts of leading zeros.

## Input

The first line contains the equation in the form A=S.

A and S will both be positive integers without leading zeros. They will be different.

A will contain **at most 1000 digits**.

S will be **less than or equal to 5000**.

**Note:** The input data will guarantee that a solution, although not necessarily unique, will always exist.

## Output

Output the corrected equation. If there are multiple solutions, output any of them.

## Examples

| **input** | **input** | **input** |
|---|---|---|
| 143175=120 | 5025=30 | 999899=125 |
| **output** | **output** | **output** |
| 14+31+75=120 | 5+025=30 | 9+9+9+89+9=125 |