Tournament Planning (poker)

Giorgio is so confident in his poker skills that he is planning his participation in the next N tournaments *based on the assumption that he will always win.*[1] However, the plan is not trivial since some tournaments may overlap, and others may be too expensive to participate in!

More precisely, each tournament i for $= 0 \dots N-1$ is held in day D_i, starts at time S_i, ends at time E_i, has a buy-in[2] B_i and a prize P_i. Before the start of the tournaments, Giorgio was able to borrow M euros from friends to spend on buy-ins.

Now, Giorgio has to select a subset of the tournaments such that for each i among them:

Figure 1: Giorgio practicing his infallible poker skills.

- every other selected tournament j on the same day does not overlap (i.e., either $S_i \geq E_j$ or $S_j \geq E_i$);
- Giorgio is able to pay for the buy-in B_i, assuming to have collected all the prizes for the selected tournament j finished before the start of i.

Of course, the selection should allow Giorgio to earn the most euros possible. Help Giorgio plan his next holidays by computing the amount of euros he will have (at most) after all the tournaments!

> ☞ Among the attachments of this task you may find a template file `poker.*` with a sample incomplete implementation.

Input

The first line contains the integers N, M: the number of tournaments and the "starting" money. Each of the following N lines contains five integers D_i, S_i, E_i, B_i, P_i: a description of the i-th tournament.

Output

You need to write a single line with an integer: the maximum amount of euros that Giorgio can have after all the tournaments.

Constraints

- $1 \leq N, M \leq 100\,000$.
- $0 \leq B_i \leq P_i \leq 10^9$ for each $i = 0 \dots N-1$.
- $0 \leq D_i \leq D_{i+1} \leq 1000$ for each $i = 0 \dots N-2$.
- $0 \leq S_i < E_i \leq 1000$ for each $i = 0 \dots N-1$.

[1] This confidence has absolutely nothing to do with secret cheating tactics.
[2] The *buy-in* of a poker tournament is the participation fee required to access it.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
- **Subtask 2** (10 points) There is at most one tournament every day.
- **Subtask 3** (15 points) $S_i = 0$ and $E_i = 1$ for each i.
- **Subtask 4** (25 points) $B_i = 0$ and $P_i = 1$ for each i.
- **Subtask 5** (20 points) $N \leq 10$.
- **Subtask 6** (15 points) $N \leq 1000$.
- **Subtask 7** (15 points) No additional limitations.

Examples

input	output

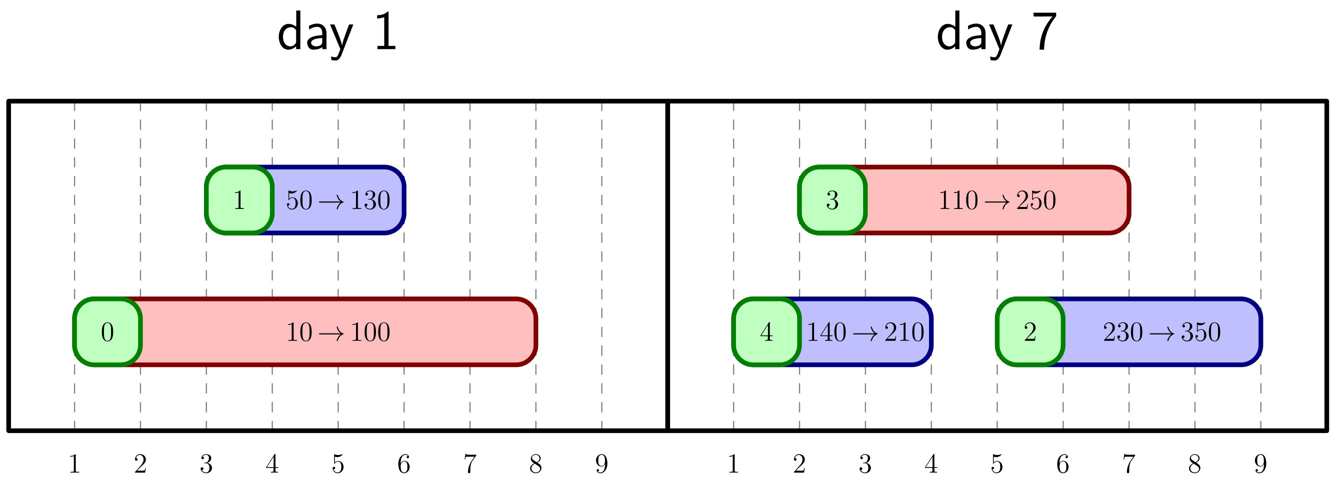
`3 100`
`2 0 10 100 300`
`3 1 8 500 1000`
`5 0 12 300 600`
	`600`

`5 50`
`1 1 8 10 100`
`1 3 6 50 130`
`7 5 9 230 350`
`7 2 7 110 250`
`7 1 4 140 210`
	`280`

Explanation

In the **first sample case**, the tournaments are all non-overlapping. By participating in the first tournament, Giorgio can have 300 euros by the end of day 2. Unfortunately, this is not enough to participate in the second tournament; it is however enough for the third tournament, at the end of which Giorgio will have 600 euros.

In the **second sample case**, the tournaments are arranged as in the following schedule.



In day 1, the best option is choosing tournament 0, at the end of which Giorgio will have $50-10+100 = 140$ euros. Even though tournaments 4 and 2 are non-overlapping in day 7, Giorgio cannot do both of them, since after tournament 4 he will not have enough money for the buy-in of tournament 2. Thus, he can just decide between tournament 4 and 3, among which the most convenient is number 3 for a final amount of $140 - 110 + 250 = 280$ euros.