Map of Tasks (tasks)

Giorgio's high-school in Italy is *serious* about the IIOT: all its the students are training for the next edition of the contest, and they absolutely want to earn their school a medal. The head Computer Science professor of the school has prepared a detailed training program for his students: a map of all the tasks that should be fully solved in order to be ready for the competition.

The map is given as a rooted tree with N nodes (numbered from 0 to $N-1$) where each node represents a task, and the parent-child relationships represent restrictions in the order in which the tasks should be solved. More specifically: if a task u has a task v among its children, this means that the task u should be solved first in order to "unlock" the task v.

The students know, for each task u, the time T_u (in hours) that it would take a single student to solve the task with no external help. Because of how the map is made, these tasks can be attempted in parallel: multiple tasks can be attempted at the same time by different students, however it's not possible for multiple students to work on the same task.

Help the students calculate the minimum amount of hours that it will take them to solve all the tasks in the map. But wait! To make things more interesting, the professor agreed that the students can "cheat" a maximum of C times: this means that for C tasks they will be able to simply find a solution online, spending 0 hours on those tasks.

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

Input

The first line contains two integers N and C, the total number of tasks in the map and the number of tasks that students are allowed to solve by “cheating”. The next N lines describe the tasks: the i-th line (with i going from 0 to $N-1$) contains two integers P_i and T_i, which indicate that:

- Before solving task i, a student must solve task P_i. If $P_i = -1$, then task i is the root of the map and it’s the first task to solve.
- Solving task i without any help will take a student T_i hours.

Output

You need to write a single integer: the minimum number of hours that the students will need to solve all the tasks.

Constraints

- $1 \leq N \leq 10\,000$.
- $0 \leq C \leq 100$.
- $0 \leq T_i \leq 10^9$ for each $i = 0 \ldots N-1$.
- $P_i = -1$ for exactly one value of i, and $0 \leq P_i < N$ for all other values of i.
- Following the $i \to P_i \to P_{P_i} \to \ldots$ chain always leads to the root of the tree, for each $i = 0 \ldots N-1$.
- The school has enough students to tackle as many tasks in parallel as needed.

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** (18 points) $N \leq 15$.

– **Subtask 3** (9 points) $C = 0$.

– **Subtask 4** (23 points) $C = 1$.

– **Subtask 5** (29 points) $C \leq 20$.

– **Subtask 6** (21 points) No additional limitations.

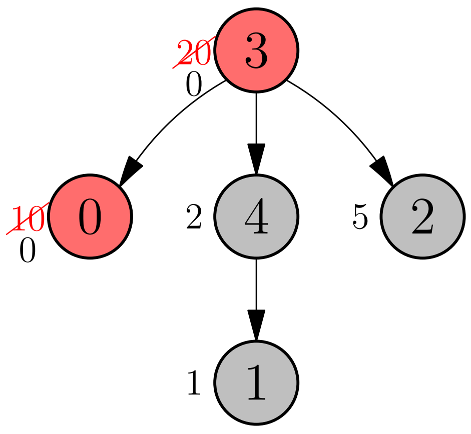
Examples

input.txt	output.txt

5 2
3 10
4 1
3 5
-1 20
3 2
	5

7 2
1 10
-1 1
3 10
1 3
1 7
0 7
5 9
	14

Explanation

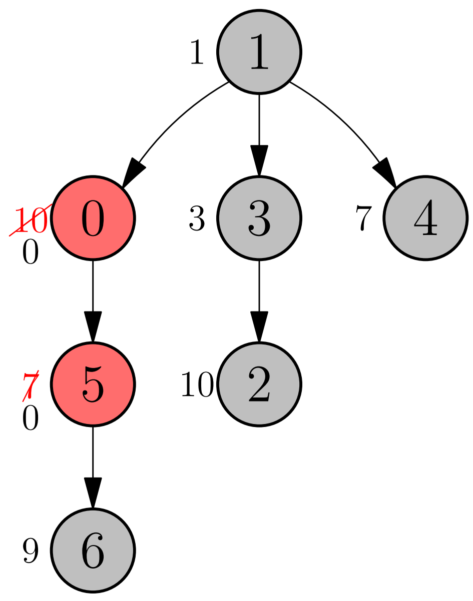


The **first sample case** is shown on the left: there are 5 tasks and only 2 of them can be *skipped*.

Skipping task 0 and task 3 leads to the optimal solution, whose total time is 5 hours. After skipping task 3 (the root), task 4 and 2 can be tackled in parallel; after 2 hours task 4 is solved and task 1 can be started.
After 5 hours task 2 is solved and you cannot choose another set of tasks to skip to reduce this time.

The **second sample case** has 7 tasks and only 2 of them can be *skipped*.

A possible optimal solution is shown on the right, reducing the total time down to 14 hours.



1 1
~~10~~ 0 0
3 3
7 4
~~7~~ 0 5
10 2
9 6