The `short` shank; Redemption (prison)

Well, that certainly didn't go as planned: While you managed to avoid the museum's watchmen, you somehow missed that there were surveillance cameras all over the place. As a consequence, your debut as an art thief turned out to be a total disaster: You were arrested and sentenced to prison.

Because WiFi access in prison is *the worst*, you want to escape. From many prison dramas, you know that distractions are crucial to any escape plan. Therefore, you decided to set up a prison rebellion as a ploy.

The N prison cells are aligned along a single hallway. The leftmost cell is number 1, the cell immediately to its right is number 2, and so on. Unfortunately, not all your fellow prisoners join the rebellion as planned. But you could make sure that the prisoner in cell i is planning to start rebelling at time t_i, i.e. exactly t_i seconds from now. Also, whenever a prisoner hears the prisoner *immediately to his left* rebelling, he will also start rebelling after just one second (unless he is already rebelling, of course). A prisoner who has started rebelling will never stop.

At time T, the local F.R.U.IT.T.Ar.T.* representative will arrive for an inspection. You figure that this will be the perfect time to escape. However, the prison wardens won't be happy about a rebellion during an inspection. You are afraid they are going to use D soundproof mattresses that can be installed between cells. If such a mattress is installed between cell $i - 1$ and cell i, the prisoner in cell i will not start rebelling before time t_i regardless of the actions of the prisoner in cell $i - 1$ (or any other prisoner).

Write a program that helps you estimate your chances of success: The program shall compute the minimal number of prisoners who will rebel at time T if the wardens place the mattresses optimally.

Input

The first line of input contains three integers N, D, and T: the number of prison cells, the number of mattresses, and the time the representative arrives.

The second line contains N integers t_i, where t_i describes the time the prisoner in cell i is planning to start rebelling.

Output

The only line of output should consist of exactly one integer: the minimal number of prisoners rebelling at time T.

Constraints

We always have $1 \le N, D \le 2\,000\,000$, $1 \le T \le 10^9$, and $1 \le t_i \le 10^9$ for all $i = 1, \ldots, N$. In addition, $D < N$ in all testcases.

Subtask 1 (15 points). $N \le 500$

Subtask 2 (10 points). $N \le 500\,000$, $D = 1$

Subtask 3 (20 points). $N \le 4\,000$

* The **F**ederal **R**egistry for **U**nhappy **IT** experts **T**urned **Ar**t **T**hieves

Subtask 4 (10 points). $N \leq 75\,000$, $D \leq 15$

Subtask 5 (25 points). $N \leq 75\,000$

Subtask 6 (20 points). No further constraints.

Samples

Input	Output
5 1 42 13 37 47 11 42	4
5 2 5 1 9 4 6 7	2

An optimal solution to the first sample case would be the following: place a mattress between the second and the third cell. In this case the first, second, fourth, and fifth prisoner will rebel.

Limits

Time: 1.5 s
Memory: 512 MiB

The Collection Game (swaps)

Phew! You only narrowly escaped from prison after your disastrous debut as an art thief. A legal route into the art business seems to be a better idea after all. So, you decided to work as an art critic at the same museum where you were caught before.

This means that you visit the museum several times and produce an art review for each visit. An art review covers several pairs of rooms of the museum. For each pair, you compare the art of the two rooms during your visit and determine which room displays the art collection with the higher aesthetic value.* In the end, however, you also want to survey the art in the entire museum. That is, using all your reviews, you want to rank all rooms of the museum by decreasing aesthetic value of the art collections displayed.

Because planning is the key, you decide in advance for each visit which pairs of rooms you will compare. Also, for the sake of diversity, no room should appear more than once in a single art review.

Unfortunately, your newly gained reputation in the art community proves to be an obstacle. Whenever you announce that you will compare a particular pair of rooms during your next visit, the museum might spontaneously swap the art collections of these two rooms. Your final survey and room ranking is supposed to be based on what is displayed in each room during your last visit.

Write a program that schedules no more than V visits to the museum and, based on the resulting art reviews, computes a list of all rooms of the museum ordered by decreasing aesthetic value of the art collections displayed in each room at the time of your last visit.

Communication

This is a communication task. You must implement the function **void** *solve*(**int** N, **int** V) where N is the number of rooms of the museum, numbered 1 to N, and V is the maximal number of visits you are allowed to make. For each testcase, this function is called exactly once. In the function, you can use the following other functions provided by the grader:

- **void** *schedule*(**int** i, **int** j) schedules a comparison of rooms i and j during your next visit ($1 \le i, j \le N, i \ne j$). Immediately after a call to this function, the museum can decide to swap the art collections of rooms i and j.
- **vector**⟨**int**⟩ *visit*() visits the museum and performs all scheduled comparisons. This function returns an array with exactly one entry for each scheduled comparison since your last visit to the museum (that is, for each call to *schedule* since the last call to *visit* or the beginning of the program). The entry at index k is 1 if the art in room i has a higher aesthetic value than in room j, and 0 otherwise. Here, i and j are the rooms from the $(k+1)$-th scheduled comparison.
- **void** *answer*(**vector**⟨**int**⟩ r) publishes your list of all the rooms of the museum ordered by decreasing aesthetic value. r must be an array of length N; its i-th entry is the room with the $(i+1)$-th most aesthetic art collection during your last visit to the museum. You must call *answer* exactly once; your program will be automatically terminated afterwards.

If any of your function calls does not match the above format, if a room is passed more than once as a parameter to *schedule* between two calls to *visit*, or if you call *visit* more than V times, your program will be immediately terminated and judged as **Not correct** for the respective testcase. You must not write anything to standard output, otherwise you may receive the verdict **Security violation!**.

* Of course, any judgement of art can only be relative. That's why, after your visit, you will only know the relative order for each pair of rooms you compared, but no order between visited rooms that were in different pairs.

If you use C++, you must include the file `swaps.h` in your source code. To test your program locally, you can link your program with `sample_grader.cpp` which can be found in the attachment for this task in CMS (see below for a description of the sample grader). The attachment also contains a sample implementation with additional explanations as `swaps_sample.cpp`.

If you use Python, you can find a sample implementation as `swaps_sample.py` in the attachment which also describes the interface for Python submissions.

Constraints

We always have $1 \le N \le 500$ and $50 \le V \le 5\,000$.

Subtask 1 (5 points). $V = 5\,000$ and the museum never swaps art collections.

Subtask 2 (10 points). $V \ge 1\,000$ and the museum never swaps art collections.

Subtask 3 (5 points). $N \le 100$, $V = 5\,000$

Subtask 4 (15 points). $V = 5\,000$

Subtask 5 (15 points). $V \ge 500$

Subtask 6 (35 points). $V \ge 100$

Subtask 7 (15 points). $V \ge 50$

Moreover, the following holds: In each of the subtasks 3 to 7 you get 60% of the points awarded for the respective subtask if you solve all testcases in which the museum always puts the art collection with the higher aesthetic value into room i for each call to *schedule*. Inside CMS this is shown as "Group 1" of the corresponding subtask.

Sample Interaction

Consider a testcase with $N = 4$ and $V = 50$ where initially the rooms are ordered $1, 2, 3, 4$ by decreasing aesthetic value. At the beginning, the grader calls your function *solve* as *solve*$(4, 50)$. Then, one possible interaction between your program and the grader could look as follows:

Your program	Return value	Explanation
schedule$(1, 2)$		schedules to compare the art in rooms 1 and 2
schedule$(3, 4)$		schedules to compare the art in rooms 3 and 4
		the museum swaps the art of rooms 3 and 4
visit()	$\{1, 0\}$	visits the museum and performs all comparisons; the art in room 1 and 4 has a higher aesthetic value than in room 2 and 3 respectively
schedule$(2, 4)$		schedules to compare the art in rooms 2 and 4
visit()	$\{1\}$	visits the museum; the art in room 2 has a higher aesthetic value than in room 4
answer$(\{1, 2, 4, 3\})$		you are convinced that the rooms are ordered $1, 2, 4, 3$ by decreasing aesthetic value
		the solution is correct and is accepted

Note that the above queries are of course not sufficient to determine the order of the rooms with certainty: for example, the order $2, 1, 4, 3$ is also consistent with all answers to *visit*. This order is

obtained when starting from $4, 1, 2, 3$ and swapping the art of rooms 2 and 4 after the last call to *schedule*.

Grader

The sample grader expects on standard input the numbers N and V as well as a list of N integers, the rooms ordered by decreasing aesthetic value at the beginning of *solve*. Then, the grader writes to standard output a protocol of all grader functions called by your program. Whenever *schedule*(i, j) is called, the grader expects on standard input the number 1 if the art collections of rooms i and j should be swapped at that point in time, or 0 otherwise. At the end, the grader writes one of the following messages to standard output:

Invalid input. The input to the grader via standard input was not of the above format.

Invalid schedule. The function *schedule* was called with invalid parameters.

Out of visits. The function *visit* was called more than V times.

Invalid answer. The function *answer* was called with invalid parameters.

Wrong answer. The function *answer* was called with a wrong list of rooms.

No answer. The function *solve* terminated without calling *answer*.

Correct: *v* visit(s) used. None of the above cases occurred and the function *visit* was called v times.

In contrast, the grader which is used for the evaluation of your submission will only output **Not correct** (for any of the above errors) or **Correct**. Both the sample grader and the grader used for evaluation will terminate your program automatically whenever one of the above errors occurs or if your program calls *answer*.

Limits

Time: 1.5 s
Memory: 512 MiB

The Xana coup (xanadu)

Having to visit the museum day after day as an art critic—and not being allowed to touch the exhibits!—turned out to be too much for you. Therefore, you decided to give your career as an art thief another try.* However, after the total disaster of your debut, you are determined to do something about the surveillance cameras this time.

Accordingly, you have used your IT skills to hack into the camera control system. Unfortunately, the cameras themselves are part of the recent installation artwork *Xanadu*. This leads to a rather strange behaviour: There are N cameras (numbered $1, \ldots, N$) distributed over the museum, some of which might already be turned off for artistic reasons. The N cameras are connected by $N - 1$ wires in such a way that any two of them are connected to each other either directly or indirectly. The camera control system offers a button for each individual camera. However, pressing such a button does not only toggle this single camera, but also *all cameras directly connected to it*.†

You are worried that your hacking effort might get noticed if you interact with the camera control system too much. Write a program that calculates the *minimal number* of button presses necessary to switch off all cameras.

Input

The first line contains an integer N, the number of cameras in the museum.

Each of the following $N - 1$ lines contains two integers a and b ($1 \le a, b \le N$, $a \ne b$). This means that cameras a and b are directly connected by a wire.

The last line contains N integers. The i-th of these integers is 1 if camera i is turned on at the beginning, and 0 if camera i is turned off.

Output

Your program should output a single line. This line should consist of an integer, the minimum number of button presses required to turn off all cameras, or the string `impossible` if it is not possible to turn off all cameras.

Constraints

It always holds that $3 \le N \le 100\,000$.

Subtask 1 (5 points). $N \le 20$

Subtask 2 (15 points). $N \le 40$

Subtask 3 (10 points). Cameras A and B are directly connected if and only if $|A - B| = 1$.

Subtask 4 (40 points). Every camera is directly connected to at most 3 other cameras.

Subtask 5 (30 points). No further constraints.

---

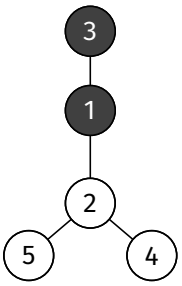
* Plus, you noticed the amazing synergies between your day job as an art critic and your night time activities as an art thief—like being able to scout the area without raising suspicion, or increasing the prizes for the art you are going to steal by publishing glowing reviews beforehand.

† It's a metaphor for how our own well-being and mood affects the well-being of those closest to us. *Obviously.*

Samples

Input	Output
5 1 2 1 3 2 4 2 5 0 1 0 1 1	4
5 1 2 2 3 3 4 4 5 0 1 1 1 1	impossible

The following graphic shows the first sample:



An optimal sequence of button presses to turn off all the cameras is given by pressing the buttons for cameras 4, 5, 3, and 1 in this order.

Limits

Time: 1 s
Memory: 512 MiB