Parcel Post (multihop)

In order to deliver parcels more efficiently, the Post Office has constructed a network of pneumatic tubes beneath the streets of London. The network consists of N routing stations connected by $N-1$ undirected tubes. There is a unique path between any pair of stations, and parcels will be sent along the path from their source to their destination.

When a parcel is at routing station i, there are two options for sending it on towards its destination. It can be fired at *low power* at a cost of A_i, in which case it will travel along a single tube to the next station along its route. Alternatively, it can be fired at *high power*. In this case, the operator will select a dial setting $k \geq 1$, and the parcel will travel along the next k tubes of its route, at a cost of $B_i + k \cdot C$.

The Post Office will route parcels so as to minimise total cost, but in order to avoid congestion on the network parcels must stay on the direct route from their source to their destination. Your task is to find the minimum costs for a series of Q parcels to be sent in this network.

Implementation

You will have to submit a single `.cpp` source file.

☞ Among this task's attachments you will find a template `multihop.cpp` with a sample implementation.

You have to implement the following functions:

C++	`void init(int N, int C, vector<int> A, vector<int> B, vector<int> U, vector<int> V);`
C++	`long long query(int X, int Y);`

- Integer N represents the number of routing stations.
- Integer C represents the incremental cost per unit of power when firing at high power, as described above.
- The array A, indexed from 0 to $N-1$, contains the cost of firing at low power from each node.
- The array B, indexed from 0 to $N-1$, contains the basic cost of firing at high power from each node, as described above.
- The arrays U and V describe the tubes in the network: there is a tube between routing station $U[i]$ to routing station $V[i]$.
- `query` should return the minimum cost of sending a parcel from routing station X to routing station Y.

The grader will call the function `init`, and then will call `query` Q times, printing its return value to the output file.

Sample Grader

The task's directory contains a simplified version of the jury grader, which you can use to test your solution locally. The simplified grader reads the input data from `stdin`, calls the functions that you must implement, and finally writes the output to `stdout`.

The input is made up of $N+Q+2$ lines, containing:

- Line 1: the integers N, Q, C.

- Line 2: the integers A_i, separated by space.
- Line 3: the integers B_i, separated by space.
- Line $4 + i$ ($0 \leq i < N - 1$): the integers U_i, V_i.
- Line $4 + (N - 1) + i$ ($0 \leq i < Q$): the integers X_i, Y_i.

The output is made up of Q lines, containing the values returned by the function `query`.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq Q \leq 100\,000$.
- $1 \leq C \leq 1\,000\,000\,000$.
- $1 \leq A_i \leq 1\,000\,000\,000$ for each $i = 0, \ldots, N - 1$
- $1 \leq B_i \leq 1\,000\,000\,000$ for each $i = 0, \ldots, N - 1$
- $0 \leq U_i < N$.
- $0 \leq V_i < N$.

Scoring

Your program will be tested on a set of test cases grouped by subtask. To obtain the score associated to a subtask, you need to correctly solve all the test cases it contains.

- **Subtask 1** [**0 points**]: Sample test cases.
- **Subtask 2** [**5 points**]: $A_i \leq 10$, $B_i \leq 10$ for each $i = 0, \ldots, N - 1$, $C \leq 10$, $N \leq 10$, $Q \leq 10$.
- **Subtask 3** [**10 points**]: $N \leq 5000$, $Q = 1$.
- **Subtask 4** [**25 points**]: $N \leq 100\,000$, $Q = 1$.
- **Subtask 5** [**25 points**]: $N \leq 5000$.
- **Subtask 6** [**35 points**]: No additional constraints.

Examples

stdin	stdout

5 1 4
2 8 6 9 2
2 5 9 5 2
3 0
2 3
4 2
1 4
0 1
	16

5 5 3
9 7 9 4 5
5 10 8 9 7
4 3
0 4
2 0
1 2
4 0
3 1
0 3
3 0
1 4
	5
	20
	11
	9
	19

Explanation

In the **first sample case**, we can fire the parcel from routing station 0 to 4 at high power, for a cost of 14, and then from 4 to 1 at low power, for a cost of 2. The final cost is then 16, which is optimal.