SheepDev Contest #1

===

A - Tree

problem

You are given a tree with N vertices, numbered from 0 to N - 1.
Each vertex u is painted color C[u].

For each vertex u, your work is to find a vertex v which satisfies these qualities.

 - v is contained in the shortest path between 0 and u
 - u is the only vertex in the shortest path between v and u which has color C[u].

If there are multiple possible v, print the one farthest from u.

input

first line - 1 <= N <= 1000000
second line - p[1], p[2], ..., p[N - 1] where p[i] is parent of vertex i, it is guaranteed that p[i] < i
third line - C[0], C[1], ..., C[N - 1]
               where 0 <= C[i] <= 10000000

output

print one line containing N integers - answer for each vertex in order.

scoring

28 points: N <= 5000
18 points: p[i] = 0 forall 1 <= i < N
18 points: p[i] = i - 1 forall 1 <= i < N
36 points: no additional constraints

sample input#1

```
2
0
4636868 4636868
```

sample output#1

```
0 1
```

sample input#2

```
10
0 1 0 0 4 0 5 7 2
1450865 1450865 479954 3564781 3564781 4808789 4808789 1450865
1450865 2458303
```

sample output#2

```
0 1 0 0 0 0 0 4 8 0
```

SheepDev Contest #1
==

B – A-K Query

problem
You have an array A[] of size N, initially filled with zeros.
You must process Q queries, each of which being either type 1 or type 2.

- "1 l r k" : assign A[i] to k for all i in [l, r] (inclusive).
- "2 l r k" : print the k-th minimum value in A[l..r].
  that is, take all A[i] for i in [l, r] and sort them nondecreasingly, then print the k-th element in sorted list. (Note that the first minimum is considered 1-th minimum)

input

first line: two integers **N Q** where 1 <= N, Q <= 100000
the next Q lines: "op l r k" where op is type of the query
0 <= l <= r < n
if op = 1 then 1 <= k <= 100000
if op = 2 then 1 <= k <= r - l + 1

output

for each type 2 query, print the result as a single integer in a line.

scoring

20 points: 1 <= N, Q <= 10000
30 points: If i < j and j-th query is of type 1, then the i-th query is also of type 1
50 points: No additional constraints.

sample input#1

```
5 6
2 1 3 2
2 1 4 3
1 0 3 23063
1 3 3 29597
2 0 3 3
1 3 3 64658
```

sample output#1

```
0
0
23063
```

sample input#2

```
10 10
1 2 9 27360
1 9 9 24922
1 5 9 54454
1 5 7 49240
1 2 2 46975
2 4 8 4
1 2 8 4169
2 1 5 5
2 0 3 1
2 0 8 1
```

sample output#2

```
49240
4169
0
0
```

SheepDev Contest #1
===

C – Suitable Weights

problem
You are given a graph with N vertices and M edge (parallel edges and loops are possible).
Each edge has an integer weight not exceeding 1000000000.
The vertices are numbered 0 to N – 1.

You must process Q queries, each in form of pair "x y". You shall imagine a situation where every edges with weight less than x or more than y is removed from the graph, then print the number of connected component in the graph.

Note that you are only imagining it, and no edges actually get removed. (That is, queries are independent of each other and deletions are not persistent).

input

first line*:* **N M Q** where 1 <= N, M, Q <= 100000
the next M lines: **u v w** denoting an edge between vertex u and v having weight w
the next Q lines: **x y** denoting the query.

    1 <= w, x, y <= 1,000,000,000
    0 <= u, v < N

output

for each query, print the answer in its own line.

scoring

20 points: N, M, Q <= 3000
30 points: for all queries, x = 1

50 points: No additional constraints.

sample input#1

```
5 5 10
1 2 780694274
2 0 648643178
0 1 118132156
3 4 206940035
2 2 171790452
3271174 982261622
165435198 642943925
188125401 973025383
12190489 723186866
408333685 783663127
8729489 956653213
33919641 943244064
88914454 893634371
195769421 710040496
540619863 867872748
```

sample output#1

```
2
4
2
2
3
2
2
2
3
3
```

sample input#2

```
5 5 10
```

```
1 1 602610850
4 2 349829920
1 3 284007095
3 1 772135348
3 2 96080488
195512440 599999949
47700015 716032666
101102642 468889515
163700712 939358125
43991788 607824664
65195355 714310237
5458899 980718592
157183626 593971636
100293488 979589154
200956097 645302487
```

sample output#2

```
3
2
3
3
2
2
2
3
3
3
```