Movers

`movers.(c|cpp|py), Movers.(java|kt)`

Department of successful computing in Tomorrow Programming School is famous for stockpiling large quantities of desks and monitors in their labs, it serves well to the whole community.

Any time a meeting is being held in a lab, the number of desks and monitors in it should be sufficient to serve all participants. In case of necessity, additional desks and/or monitors may be borrowed from neighbouring labs. In extreme cases, all desks and all monitors from all neighbouring labs may be brought in. Thus, available desks and monitors in a lab are exactly those desks and monitors which are in the lab itself and in all its neighbour labs. Desks or monitors are never transported from more distant labs, it is deemed ineffective and accident prone. After a meeting, all borrowed desks and monitors are returned back to their original labs, before any other meeting starts.

The desired configuration for a meeting is when the number of available desks and monitors in the lab is equal. Often, the number of available desks in a lab is either smaller or bigger than the number of available monitors, and that creates specific problems for the maintenance staff each time.

The inflow of desks and monitors to the department is rapid. Frequently a shipment of desks and monitors arrives to the department and its contents is added to various labs. It is added immediately or immediately after the end of the current meeting.

To plan the meetings, and equipment maintenance as well, it is important to know how many desks and monitors are available in any lab at any moment. The department needs a program that can process two kinds of queries. The first kind of query specifies a number of desks or monitors that have just been added to a particular lab. The second kind of query asks for a relation between the number of available desks and monitors in a particular lab.

Input Specification

The first input line contains three integers N, M, Q ($1 \le N \le 10^5, 0 \le M \le 10^5$, $0 \le Q \le 10^5$), the number of labs, the number of pairs of labs which are neighbours to each other, and the number of queries. Labs are labeled by integers $1, 2, \ldots, N$. The second line contains N space-separated integers D_i ($0 \le D_i \le 100$), the number of desks in lab i. The third line contains N space-separated integers E_i ($0 \le E_i \le 100$), the number of monitors in lab i. Next M lines contain space-separated unique pairs of distinct integers a_i, b_i, ($1 \le a_i, b_i \le N$), the labels of pairs of neighbour labs. Next Q lines contain the queries, one query per line. Each query is provided in one of the two formats.

1. `add <count> desk/monitor <label>` – query increases the number of desks or monitors by the given `<count>` in a lab with specified `<label>`.
2. `check <label>` – query asks for a relation between the number of available desks and available monitors in a lab with specified `<label>`.

One `add` query increases the number of desks or monitors in a lab by at most 100.

Output Specification

Output Q lines, one for each query of type `check`, in the order they appear in the input. Each line contains one of the strings "`desks`", "`monitors`", or "`same`", depending on whether there are more available desks or more available monitors in the lab specified by the query, or whether their numbers are equal.

Sample Input 1

```
4 5 8
1 1 1 0
2 0 2 0
1 2
2 3
3 4
4 1
1 3
check 2
add 2 desk 1
check 2
add 1 monitor 3
check 1
check 2
check 3
check 4
```

Output for Sample Input 1

```
monitors
desks
same
same
same
monitors
```