Some time ago Mirko founded a new tourist agency named "Dreams of Ice". The agency purchased N icy islands near the South Pole and now offers excursions. Especially popular are the emperor penguins, which can be found in large numbers on the islands.

Mirko's agency has become a huge hit; so big that it is no longer cost-effective to use boats for the excursions. The agency will **build bridges** between islands and transport tourists by buses. Mirko wants to introduce a computer program to manage the bridge building process so that fewer mistakes are made.

The islands are numbered 1 through N. No two islands are initially connected by bridges. The initial number of penguins on each island is known. That number may change, but will always be between 0 and 1000 (inclusive).

Your program must handle the following three types of commands:

- "bridge A B" – an offer was received to build a bridge between islands A and B (A and B will be different). To limit costs, your program must accept the offer **only if there isn't already a way** to get from one island to the other using previously built bridges. If the offer is accepted, the program should output "yes", after which the bridge is built. If the offer is rejected, the program should output "no".
- "penguins A X" – the penguins on island A have been recounted and there are now X of them. This is an informative command and your program does not need to respond.
- "excursion A B" – a group of tourists wants an excursion from island A to island B. If the excursion is possible (it is possible to get from island A to B), the program should output the **total number of penguins** the tourists would see on the excursion (including islands A and B). Otherwise, your program should output "impossible".

Important note: your program must output responses to commands "bridge" and "excursion" immediately after they are received. The server program will not send the next command until your program responds to the previous one.

Another important note: for the server program to be able to read your program's responses, your program must flush the standard output after every response it outputs.

- In C++, use the command `cout << flush;`
- In C, use `fflush(stdout);`
- In pascal, use `flush(output);`

INPUT

The first line contains the integer N ($1 \leq N \leq 30000$), the number of islands.

The second line contains N integers between 0 and 1000, the initial number of penguins on each of the islands.

The third line contains an integer Q ($1 \leq Q \leq 300000$), the number of commands.

Q commands follow, each on its own line. As noted above, after receiving a command "bridge" or "excursion", your program will not receive another command until it has responded to the previous one.

OUTPUT

Output the responses to commands "bridge" and "excursion", each on its own line.

SCORING

In test cases worth 50% of points, the command "penguins" will not appear. In these test cases N will be odd, while in all other cases N will be even.

EXAMPLES

input

```
5
4 2 4 5 6
10
excursion 1 1
excursion 1 2
bridge 1 2
excursion 1 2
bridge 3 4
bridge 3 5
excursion 4 5
bridge 1 3
excursion 2 4
excursion 2 5
```

output

```
4
impossible
yes
6
yes
yes
15
yes
15
16
```

input

```
6
1 2 3 4 5 6
10
bridge 1 2
bridge 2 3
bridge 4 5
excursion 1 3
excursion 1 5
bridge 3 4
excursion 1 5
penguins 3 10
excursion 1 3
bridge 1 5
```

output

```
yes
yes
yes
6
impossible
yes
15
13
no
```