7669 Binary Tree

You may consider it weird, but I like trees.

Not the green ones, of course.

Among all kinds of trees, I favor binary trees.

Why?

Don't be so serious.

I always play with binary trees.

Recently, I defined a new value function on weighted binary trees.

I denoted the weight of node v as $w[v]$.

Then I defined the value of node v, $f[v]$, recursively as $w[v] + f[l] - f[r]$, where l is the left child and r is the right child of node v.

If the left child of node v does not exist, replace $f[l]$ by 0.

Similarly, if the right child of node v does not exist, replace $f[r]$ by 0.

And I do some operations to have fun:

1. Make node u be the left child of node v.
2. Make node u be the right child of node v.
3. Swap two children of node v.
4. Change the weight of node v into d.
5. Add d to the weight of all nodes which are in the subtree of node v.

I want to know the value of some nodes after doing some operations.

Can you do me a favor?

Input

There are no more than 10 trees.

For each tree, the first line contains one integer n ($1 \le n \le 40000$), indicating the number of nodes of the tree. All nodes are numbered from 1 to n.

The second line contains n integers. The i-th integer indicates $w[i]$ ($-1000 \le w[i] \le 1000$), indicating the initial weight of the node i (i starts from 1).

Then n lines follow. Among them, the i-th line contains two integer $l[i]$ and $r[i]$, indicating the left child and right child of node i.

If a child doesn't exist, replace it by '0'.

It is guaranteed that node 1 is the root, and the tree is connected.

Then one line contains an integer m ($1 \le m \le 40000$), indicating the number of operations and requests.

Then m lines, and each line is an operation or a request.

There are 5 kinds of operations as shown below:

- `1` u v — Make node u be the left child of node v.
- `2` u v — Make node u be the right child of node v.

- `3` v — Swap two children of node v.
- `4` v d — Change the weight of node v into d.
- `5` v d — Add d to the weight of all nodes which are in the subtree of node v.

The request is in the following format:

`6` v

It request the value of node v after all previous operations.

Please note that: $1 \leq u, v \leq n$, $-1000 \leq d \leq 1000$

Output

For each tree, output m lines.

For an operation, output '`S`' or '`F`', indicating the operation is successful or failed.

For operation 1: Output '`F`' if node v is in the subtree whose root is node u, or if node v already has a left child. Output '`S`' otherwise.

For operation 2: Output '`F`' if node v is in the subtree whose root is node u, or if node v already has a right child. Output '`S`' otherwise.

For operation 3: Output '`S`' if node v has two children. Output '`F`' otherwise.

For operation 4: Output '`S`'.

For operation 5: Output '`S`'.

Note that node v is in the subtree whose root is v.

If an operation failed, do nothing to the tree.

For each request, output the value of node v.

Sample Input

```
5
1 2 3 4 5
2 3
4 5
0 0
0 0
0 0
16
6 2
1 2 4
1 4 3
1 4 3
6 3
2 4 3
2 4 2
6 3
6 1
3 1
3 2
6 1
4 4 -1
6 3
5 2 1
6 1
```

Sample Output

```
1
F
S
F
7
S
F
-1
-1
S
F
3
S
4
S
8
```