

Tree manager

In this problem you must perform operations on a rooted tree storing integers in each node. There are several operations to handle:

- *changeValue(x)* - Changes the value stored in the current node to x .
- *print()* - Prints the values stored in the current node.
- *visitLeft()* - Sets the current node to be the left sibling of the current node.
- *visitRight()* - Sets the current node to be the right sibling of the current node.
- *visitParent()* - Sets the current node to be the parent of the current node.
- *visitChild(n)* - Sets the current node to be the n^{th} child of the current node. Children are numbered from left to right starting from 1.
- *insertLeft(x)* - Inserts a new node with value x as the left sibling of the current node.
- *insertRight(x)* - Inserts a new node with value x as the right sibling of the current node.
- *insertChild(x)* - Inserts a new node as the leftmost child of the current node.
- *delete()* - Deletes the current node with the subtree rooted in it and sets the current node as a parent of just deleted node.

Knowing that the tree initially consists of the root with value 0, your task is to perform Q consecutive operations.

Check the *Input Format* section for a description of how each operation is given in the input, and review the *Constraints* section to clarify which operations are not allowed for the root node.

Input Format

The first line contains a single integer, Q , denoting the number of operations to perform. The Q subsequent lines each describe a single operation to perform. The operations are coded as follows:

- **change x** \rightarrow *changeValue(x)*
- **print** \rightarrow *print()*
- **visit left** \rightarrow *visitLeft()*
- **visit right** \rightarrow *visitRight()*
- **visit parent** \rightarrow *visitParent()*
- **visit child n** \rightarrow *visitChild(n)*
- **insert left x** \rightarrow *insertLeft(x)*
- **insert right x** \rightarrow *insertRight(x)*
- **insert child x** \rightarrow *insertChild(x)*
- **delete** \rightarrow *delete()*

Constraints

- $1 \leq Q \leq 10^5$
- $0 \leq x \leq 10^6$
- $1 \leq n \leq 10$
- It is guaranteed that all operations given as input will be valid.

Invalid operations are:

- Visiting left/right sibling when there is no such sibling.
- Visiting the n^{th} child when there are less than n children.
- Deleting the root.
- Inserting any sibling of the root.
- A single node will never have more than **10** children.

Output Format

For each *print()* operation, output a single line with the value in the current node.

Sample Input

```
11
change 1
print
insert child 2
visit child 1
insert right 3
visit right
print
insert right 4
delete
visit child 2
print
```

Sample Output

```
1
3
4
```

Explanation

There are **11** operations to handle.

At the beginning, we change the value stored in the root to **1** and then we print it.

After that, we insert a new child of the root with value **2**.

Then, we visit this child and insert a new node with value **3** as its right sibling.

Next, we visit the last inserted node and print its value.

After that, we insert a new node with value **4** as the right sibling of last inserted node.

Then, we delete the current node (the one with value **3**), so the current node becomes the root.

Next, we visit the second child of the root, which is the last inserted node with value **4** (because we deleted the node with value **3**).

Finally, we print the value stored in the current node, which is **4**.