PIB '20 P3 - Rulebreaker

Time limit: 0.5s **Memory limit:** 128M

John has an ongoing contest with some of his friends. The problem is that they constantly find loopholes to exploit John's contest, so John has to make some new rules, but sometimes John's friends find loopholes to those rules. Then, he'll make new rules to overrule those rules.

In essence, John has a list of rules, each being able to do two operations:

1. Create a new rule.
2. Create a new rule to not follow the previous rule that was created.

After a while, John's friends get confused about which rule to follow, and so they come to ask you for help. They will give you Q queries of one of the following forms:

- `1` Create a new rule. It is guaranteed the first query will always be of this type.
- `2` Create a new rule to not follow the previous rule that was created.
- `3 x` Output whether John's friends should follow the x^{th} rule that was created or not. It is guaranteed $1 \leq x \leq S$, where S is the number of rules that has been created so far. x is one indexed.

Note: Rules that are created later are given higher priority in determining which rules to follow.

For example, if the following queries were given:

1. Create a rule.
2. Create a rule to not follow the previous rule.
3. Create a rule.
4. Create a rule to not follow the previous rule.

In this case, rules 1 and 3 would not be followed and rules 2 and 4 will be followed.

Note: Python users are recommended to use CPython over PyPy and add the following line to the top of their solution for performance purposes: `input = sys.stdin.readline`.

Input Specification

The first line will contain the integer Q $(1 \leq Q \leq 10^5)$, the number of queries.

The next Q lines will each contain a query as defined above.

Output Specification

For each type `3` query, output `1` if John's friends should follow rule x and `0` otherwise on its own line.

Subtasks

Subtask 1 [19%]

$Q \leq 100$

Subtask 2 [81%]

No additional constraints.

Sample Input for Subtask 1

```
5
1
2
3 1
2
3 1
```

Sample Output for Subtask 1

```
0
1
```

Explanation for Sample for Subtask 1

1. Create a new rule, indexed `1`.
2. Create a new rule, indexed `2`, to not follow the previous rule (rule `1`). Currently, John's friends should follow rule `2` and not follow rule `1`.
3. Query if rule `1` should be followed - it should not. Therefore, `0` is outputted.
4. Create a new rule, indexed `3`, to not follow the previous rule (rule `2`). This means John's friends should follow rules `1` and `3` but not rule `2`.
5. Query if rule `1` should be followed - it should. Therefore, `1` is outputted.

Sample Input for Subtask 2

```
101
1
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
```

```
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
```

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

Sample Output for Subtask 2

```
0
1
```