Baltic OI '15 P2 - Editor

Time limit: 1.5s **Memory limit:** 512M
Java: 3.0s

Baltic Olympiad in Informatics: 2015 Day 1, Problem 2

Byteasar is a programmer who works on a revolutionary text editor. In the editor there are two types of operations: one type allows to *edit text* in the editor, and the other type allows to *undo* previously performed operations. One of the innovative features of this editor is a *multilevel undo operation*. It works as follows. We say that a text editing operation is an operation of level 0. An undo operation of *level* i (for $i = 1, 2, \ldots$) undoes the last operation of level at most $i - 1$ which is not undone. For instance, an undo operation of level 1 can undo only editing operations, and an undo operation of level 2 can undo editing operations as well as undo operations of level 1 (but no undo operations of greater levels).

More formally, each of the already performed operations can be in two states: **active** or **undone**. Let X be one of the operations. Just after performing the operation X, it is in the state **active**. If X is an undo operation of level i, we find the most recent operation in state active of level at most $i - 1$ (denote it by X_1) and change the state of the operation X_1 to **undone**. If X_1 is also an undo operation, we must change to **active** the state of the operation which X_1 had undone (say X_2). We continue in the same manner: whenever the state of an undo operation X_j which had previously undone some operation X_{j+1} changes, we must also change the state of the operation X_{j+1} (which, of course, may result in changing states of further operations). The whole chain of state modifications finishes when an editing operation is reached.

For simplicity, the current contents of text in the editor will be specified by a single integer s, called the *editor state* (equal to 0 at the beginning). Each editing operation specifies the editor state that it produces. The editor state depends on the last editing operation in the state **active**. Help Byteasar and write a program which keeps track of the editor state.

Let us see this in action: the following table shows some operations performed by Byteasar and the editor state after performing each of them. The symbol E_s denotes an editing operation which changes the editor state to s, whereas the symbol U_i denotes an undo operation of level i.

Operation		E_1	E_2	E_5	U_1	U_1	U_3	E_4	U_2	U_1	U_1	E_1
Editor State	0	1	2	5	2	1	2	4	2	1	0	1

First, Byteasar performed three editing operations. The editor state changed from 0 to 1, then to 2, and finally to 5. Next, he performed two undo operations of level 1, which undid the operations E_5 and E_2 (changing their state to **undone**). Thus the editor state was restored to 1. The following undo operation of level 3 undid the last operation U_1 (changing its state to **undone**), consequently restoring the operation E_2 (changing its state back to **active**). As a result the editor state changed once again to 2. Operation U_2 undid the operation E_4, operation U_1 once again undid the restored operation E_2, the last operation U_1 undid the operation E_1, and the final operation is E_1.

Input Specification

The first line of the input contains a positive integer n, specifying the number of operations performed by Byteasar. The next n lines contain descriptions of operations, one per line, each being an integer a_i $(-n \leq a_i \leq n, a_i \neq 0)$. If $a_i > 0$, then it specifies an editing operation which modifies the editor state to a_i. If $a_i < 0$, then it specifies an undo operation of level a_i. You can assume that for every undo operation there will be some operation in the state **active** of smaller level to undo.

Output Specification

Your program should output n lines. The i-th line should contain one integer specifying the editor state after performing the first i operations from the input.

Constraints

Subtask	Conditions (in each test case)	Points
1	$n \leq 5000$	20
2	$n \leq 300\,000$ and there are only operations E_i and U_1.	15
3	$n \leq 300\,000$ and only the last number in the sequence is graded (however, the first $n-1$ numbers must be integers ranging from 0 to n).	28
4	$n \leq 300\,000$	37

Sample Input

```
11
1
2
5
-1
-1
-3
4
-2
-1
-1
1
```

Sample Output

```
1
2
5
2
1
2
4
2
1
0
1
```