

Stack

Problem ID: stack2

Write your own implementation of a stack, specialized to store 32-bit signed integers. Stacks are simple and only have a few operations.

The data structure should support the following operations:

- Default construction - initializes an empty stack. This is not tested explicitly here!
- Assignment and copy construction - must properly copy the contents of another instance of the data structure. Ensure the two instances do not share memory afterwards!
- Push - must insert an element on top of the stack.
- Pop - must remove the top element off of the stack.
- Top - must provide access to the top element of the stack.
- Size - must provide the size of the stack.

You must avoid any memory leaks or other memory errors in your implementation.

Input

The input starts with a line containing an integer q , representing the number of lines that follow. Each line will represent an operation that is run. The lines have the following format: `<id> <operation> [arg]`

The instance ID will be an integer from 1 to 1,000, representing an instance of the data structure. Each instance should be default initialized at the start as an empty stack.

The operations are the following:

- `a` - construct a copy of the stack, takes integer argument for the instance ID of the stack to copy
- `+` - push a value on the stack, takes integer argument for the value to push
- `-` - pop the stack, no additional arguments
- `t` - output the top element of the stack, see output section
- `s` - output size of stack, see output section

You may assume requested operations will not cause an error in a correctly implemented data structure. For example, a pop operation on an empty stack will not occur in the input.

Output

For each top operation, output a line with the top element of the stack.

For each size operation, output a line with the size of the stack.

Sample Input 1

```
7
1 + 1
1 t
1 + 2
1 t
1 s
1 -
1 s
```

Sample Output 1

```
1
2
2
1
```

Sample Input 2

```
6
1 + -9
1 + 6
1 + 42
2 + -5
1 s
2 s
```

Sample Output 2

```
3
1
```