Heap
Problem ID: heap2

Write your own implementation of a heap, specialized to store 32-bit signed integers. Heaps only have a few operations.

The data structure should support the following operations:

- Default construction - initializes an empty heap. 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 to the heap.

- Pop - must remove the smallest element from the heap.

- Peek - must provide access to the smallest element in the heap.

- Size - must provide the size of the heap.

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 heap.

The operations are the following:

- `a` - construct a copy of the heap, takes integer argument for the instance ID of the heap to copy

- `+` - push a value to the heap, takes integer argument for the value to push

- `-` - pop the heap, no additional arguments

- `p` - output the smallest element in the heap, see output section

- `s` - output size of heap, 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 heap will not occur in the input.

Output

For each peek operation, output a line with the smallest element in the heap.

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

Sample Input 1	Sample Output 1
7 1 + 1 1 p 1 + 2 1 p 1 s 1 - 1 s	1 1 2 1

Sample Input 2

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

Sample Output 2

```
3
1
```