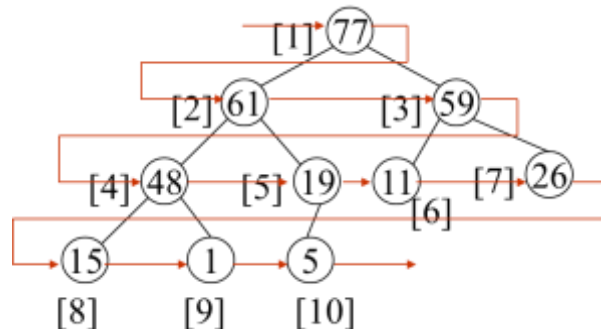


# Max Heap

Heap is a specialized tree-based data structure that satisfies the heap property: if P is a parent node of C, then the key (the value) of P is either  $\geq$  (in a max heap) or  $\leq$  (in a min heap) the key of C. The node at the "top" of the heap (with no parents) is called the root node.

In binary-tree based heap, it can be implemented via an array with property: if a node in the binary tree has index i, then its two children have index 2i, 2i+1 respectively. The following is an example of a Max-Heap:



Index	1	2	3	4	5	6	7	8	9	10
Value	77	61	59	48	19	11	26	15	1	5

There are two main operations on heap:

**Insert a new key:** *append new key to the end of the array, and then adjust the heap*

**Pop root node:** *replace the root by the last node in the array, and then adjust the heap*

			<p><b>Insert:</b> Initially, heap size is 5. We insert x in to array[6], and continue swapping the node (x) with its farther node if <math>x &gt; \text{farther.key}</math> (adjust)</p>

In this question, you need to implement the data structure of Max-Heap with functions `insert()` and `pop()`. Libraries such as `<algorithm>` `<queue>` already implements the heap (`make_heap` in `<algorithm>`, `priority_queue` in `<queue>`), therefore you are not allowed to use these libraries.

### Input

The input contains multiple test cases. Each test case begins with one integer  $n$  ( $0 \leq n \leq 100000$ ), indicating the number of operations. The following  $n$  lines give the operations on the heap, each line follows the format:

“a k”: **insert** a new number  $k$  to the heap,  $1 \leq k \leq 1000$ .

“p”: **pop** (remove) the root node of the heap.

“r”: **print** the sum of all numbers in the heap.

It will guarantee that the heap is not empty when encountered with pop operation.

### Output

For each “r” operation, print the sum in a separate line.

Sample input	Sample output
13 a 61 a 1 a 77 a 19 a 26 a 15 a 59 a 5 a 48 a 11 p p r 8 a 5 a 4 p r a 2 a 3 p r	184 4 5

In the first sample, after inserting the integers {61, 1, 77, 19, 26, 15, 59, 5, 48, 11}, we'll get the max-heap shown in the above picture. Then after two pop operations, 77 and 61 will be popped out and the sum of the remaining numbers is 184.