Given an array \( A[0..n-1] \) and an index \( i \), the \( i \)-th partial sum of \( A \) is \( \sum_{0 \leq j \leq i} A[j] \). Here, you have to implement a data structure to efficiently compute partial sums. The operations you must consider are the creation of an array with all its values initialized to zero, the modification of a value, and the query of a partial sum.

**Input**

Input consists of a non-empty sequence of commands. Every command begins with a letter to identify it, followed by one or two integer-number parameters. These are the possible commands:

- "r \( n \)" resets (or creates) an array of \( n \) integer numbers to zero. Assume \( 1 \leq n \leq 10^5 \).
- "s \( i \ x \)" sets the position \( i \) to \( x \). Assume \( 0 \leq i < n \) and \( -100 \leq x \leq 100 \).
- "g \( i \)" gets (and prints) the \( i \)-th partial sum. Assume \( 0 \leq i < n \).

In general, there are much more set and get commands than reset commands. The first command is always a reset.

**Output**

For each get command, print the corresponding partial sum. Print the output corresponding to each reset command on a unique line, separated by spaces.

**Sample input**

```
 r 8
 s 0 3  s 1 2  s 2 1  s 3 5  s 4 4  s 5 3  s 6 2  s 7 3  
 g 0  g 1  g 2  g 3  g 4  g 5  g 6  g 7
 s 3 8  
 s 3 -100  
 r 3  
 s 1 4  
 g 0  g 1  g 2  g 0
```

**Sample output**

```
3 5 6 11 15 18 20 23 6 26 3 -82  
0 4 4 0
```

**Problem information**

Author : Jordi Petit
Generation : 2013-09-02 15:37:33

http://www.jutge.org