**Java BitSet**

<https://www.hackerrank.com/challenges/java-bitset/problem>

Java's [BitSet](https://docs.oracle.com/javase/7/docs/api/java/util/BitSet.html) class implements a vector of bit values (i.e.: false (0) or true (1)) that grows as needed, allowing us to easily manipulate bits while optimizing space (when compared to other collections). Any element having a bit value of 1 is called a set bit.

Given 2 BitSets, B1 and B2, of size N where all bits in both BitSets are initialized to 0, perform a series of M operations. After each operation, print the number of set bits in the respective BitSets as two space-separated integers on a new line.

**Input Format**

The first line contains 2 space-separated integers, N (the length of both BitSets B1 and B2) and M (the number of operations to perform), respectively.  
The M subsequent lines each contain an operation in one of the following forms:

* [AND](https://en.wikipedia.org/wiki/Logical_conjunction) <set> <set>
* [OR](https://en.wikipedia.org/wiki/Logical_disjunction) <set> <set>
* [XOR](https://en.wikipedia.org/wiki/Exclusive_or) <set> <set>
* [FLIP](https://en.wikipedia.org/wiki/Bitwise_operation#NOT) <set> <index>
* [SET](https://docs.oracle.com/javase/7/docs/api/java/util/BitSet.html#set%28int%29) <set> <index>

In the list above, <set> is the integer 1 or 2, where 1 denotes B1 and 2 denotes B2.  
<index> is an integer denoting a bit's index in the BitSet corresponding to <set>.

For the binary operations AND, OR, and XOR, operands are read from left to right and the BitSet resulting from the operation replaces the contents of the first operand. For example:

AND 2 1

B2 is the left operand, and B1 is the right operand. This operation should assign the result of B2 ^ B1 to B2.

**Constraints**

* 1 <= N <= 1000
* 1 <= M <= 10000

**Output Format**

After each operation, print the respective number of set bits in BitSet B1 and BitSet B2 as  space-separated integers on a new line.

**Sample Input**

5 4

AND 1 2

SET 1 4

FLIP 2 2

OR 2 1

**Sample Output**

0 0

1 0

1 1

1 2

**Explanation**

Initially: N = 5, M = 4, B1 = {0,0,0,0,0}, and B2 = {0,0,0,0,0}. At each step, we print the respective number of set bits in B1 and B2 as a pair of space-separated integers on a new line.

M0 = AND 1 2  
B1 = B2 ^ B1 = {0,0,0,0,0} ^ {0,0,0,0,0} = {0,0,0,0,0}  
B1 = {0,0,0,0,0}, B2 = {0,0,0,0,0}  
The number of set bits in B1 and B2 is 0.

M1 = SET 1 4  
Set B1[4] to true (1).

B1 = {0,0,0,0,1}, B2 = {0,0,0,0,0}.  
The number of set bits in B1 is 1 and B2 is 0.

M2 = FLIP 2 2  
Set B2[2] from false (0) to true (1).

B1 = {0,0,0,0,1}, B2 = {0,0,1,0,0}.  
The number of set bits in B1 is 1 and B2 is 1.

M3 = OR 2 1  
B2 = B2 v B1 = {0,0,1,0,0} v {0,0,0,0,1} = {0,0,1,0,1}  
B1 = {0,0,1,0,0}, B2 = {0,0,1,0,1}  
The number of set bits in B1 is 1 and B2 is 2.