# Item Stock

Max. score: 10

Items in Shopee can have their stocks derived from other items. For example, 1 stock of item A can be derived from 2 stock of item B + 3 stock of item C. We say that item B and item C are parents of item A. For this problem, we are only interested when an item can only have 1 parent item. In this case, we can see the structure of stock derivation will form a rooted tree.

There are 2 kinds of derivations:

1. Dynamic stock derivation. Suppose that 1 stock of item A equals to Qty stock of item B. Then, the stock of item A will be equal to floor(item_B_stock / Qty).
2. Fixed stock derivation. Suppose that 1 stock of item A equals to Qty stock of item B, and we initially have S stock of item A. Then, item A will deduct stocks from its lowest ancestor which is fixed stock, to make sure that item A will have sufficient stock. It can be assumed that the root of the tree (1st item) will always be fixed stock. Note that the number of reserved stocks depends on the multiplication of the Qty from the path of item A to that ancestor, not just the Qty to item B. Please refer to the example input for clarity.

At first, we only have item 1, which initially has **M** stock. Then, we add **N**-1 items one-by-one, possibly changing the stock of some items at each step. In the end, what will be the stock of each item?

## Input

The first line contains 2 integers **N** ($1 \le N \le 100,000$) and M ($1 \le M \le 1,000,000,000$), denoting the number of items and the initial stock of the 1st item.

The next **N**-1 lines contain the description of the i-th item (starting from 2), which can be in one of the 2 following formats:

- **1** $P_i$ $Qty_i$ ($1 \le P_i < i$, $1 \le Qty_i \le 10$), which means the i-th item has dynamic stock with parent item $P_i$ and 1 stock of it equals to $Qty_i$ stock of its parent
- **2** $P_i$ $Qty_i$ $S_i$ ($1 \le P_i < i$, $1 \le Qty_i \le 10$, $1 \le S_i \le 1,000,000,000$), which means the i-th item has fixed stock with parent item $P_i$, 1 stock of it equals to $Qty_i$ stock of its parent, and has initial stock of $S_i$.

It is guaranteed that at the end, the stock for each item will be non-negative.

## Output

Output **N** lines, each containing an integer. The integer in the i-th line denotes the stock of the i-th item.

SAMPLE INPUT

```
5 15
1 1 2
2 1 3 1
1 2 1
2 4 3 1
```

SAMPLE OUTPUT

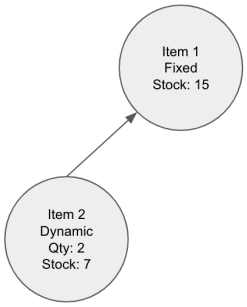
```
6
3
1
3
1
```

**Explanation**

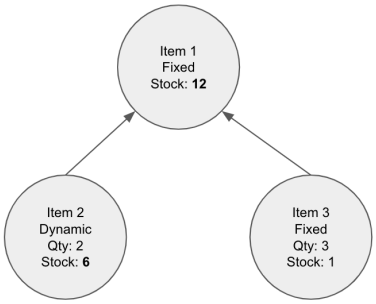
Below are the states after each item additions:

1. Initial state

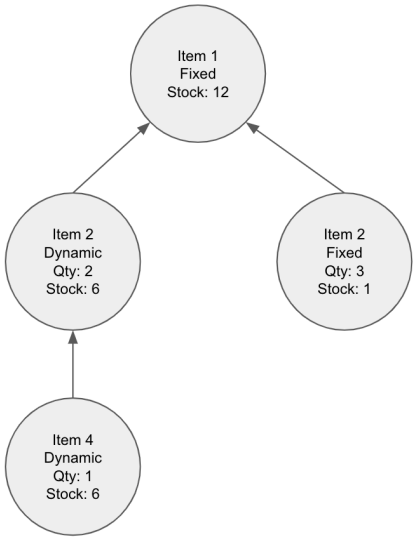

2. Adding 2nd item, stock is floor(15/2) = 7




3. Adding 3rd item, taking 1 * 3 stock from the 1st item. Note that Item 2 stock is also changed because of this.

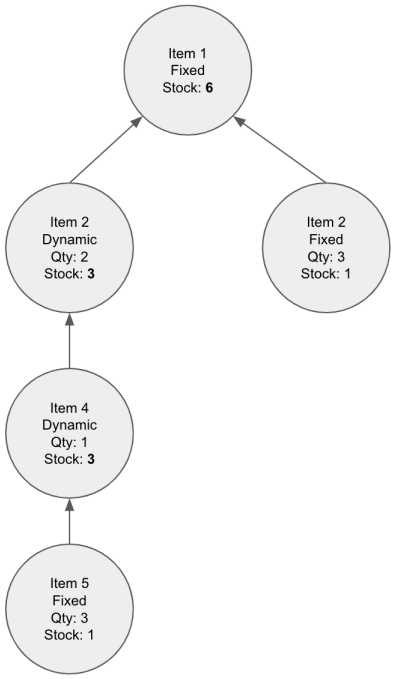


4. Adding 4th item, stock is floor(6/1) = 6






5. Adding 5th item, taking 2*1*3 (Qty) *1 (stock) stock from the 1st item as it is its lowest fixed stock ancestor. Note that Item 2 and item 4 stock are also changed because of this.




| | |
|---|---|
| **Time Limit:** | 1.0 sec(s) for each input file. |
| **Memory Limit:** | 256 MB |
| **Source Limit:** | 1024 KB |
| **Marking Scheme:** | Score is assigned when all the testcases pass. |
| **Allowed Languages:** | Bash, C, C++, C++14, C++17, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, Java 14, JavaScript(Rhino), JavaScript(Node.js), Julia, Kotlin, Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, Python 3.8, R(RScript), Racket, Ruby, Rust, Scala, Swift-4.1, Swift, TypeScript, Visual Basic |

## CODE EDITOR

Save C (gcc 5.4.0)

```c
/*
// Sample code to perform I/O:
#include <stdio.h>

int main(){
    int num;
    scanf("%d", &num);                          // Reading input from STDIN
    printf("Input number is %d.\n", num);       // Writing output to STDOUT
}

// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
*/

// Write your code here

```



Provide custom input

COMPILE & TEST SUBMIT

Tip: You can submit any number of times you want. Your best submission is considered for computing total score.

Your Rating:

View all comments