# Problem 3. Heart Delivery

*Valentine’s Day is coming, and Cupid has very limited time to spread some love across the neighborhood. Help him with his mission!*

You will receive a **string** with **even integers,** separated by a **"@".** This is our neighborhood. After that a series of **Jump** commands will follow, until you receive **"Love!"** Every house in the neighborhood needs a certain number of **hearts** delivered by Cupid, in order to be able to celebrate Valentine’s Day. Those needed hearts are indicated by the integers in the neighborhood.

Cupid starts at the position of the **first** **house** (index 0) and must jump by a **given length.** The jump commands will be in this format: **"Jump {length}"**.

Every time he jumps from one house to another, the needed hearts for the visited house are **decreased by 2**. If the needed hearts for a certain house become **equal to 0** , print on the console **"Place {houseIndex} has Valentine's day."** If **Cupid** jumps to a house where the needed hearts are **already** **0,** print on the console"**Place {houseIndex} already had Valentine's day.**".

Keep in mind that **Cupid** can have a **bigger jump length** than the **size of the neighborhood** and if he does jump **outside** of it, he should **start** from the **first house** again**.**

*For example, we are given this neighborhood: 6@6@6. Cupid is at the start and jumps with a length of 2. He will end up at index 2 and decrease the needed hearts there by 2: [6, 6, 4]. Next he jumps again with a length of 2 and goes outside the neighborhood, so he goes back to the first house (index 0) and again decreases the needed hearts there: [4, 6, 4].*

### Input

* On the first line you will receive a **string** with **even integers** separated by **"@"** –the neighborhood and the number of hearts for each house.
* On the next lines, until "**Love!**" is received, you will be getting jump commands in this format: "**Jump {length}**".

### Output

At the end print **Cupid's** **last position** and whether his mission was successful or not:

* "**Cupid's last position was {lastPositionIndex}.**"
* If **each house** has had a Valentine's day, print:
  + "**Mission was successful.**"
* If **not,** print the **count** of all houses that **didn`t** celebrate a Valentine's Day:
  + **"Cupid has failed {houseCount} places."**

### Constraints

* The **neighborhood`s** size will be in the range [1…20]
* Each **house** will need an **even number** of hearts in the range [2 … 10]
* Each **jump length** will be an integer in the range [1 … 20]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10@10@10@2  Jump 1  Jump 2  Love! | Place 3 has Valentine's day.  Cupid's last position was 3.  Cupid has failed 3 places. | Jump 1 ->> [10, 8, 10, 2]  Jump 2 ->> [10, 8, 10, 0] so we print "Place 3 has Valentine's day."  Next command is "Love!", so we print Cupid`s last position and the outcome of his mission. |
| 2@4@2  Jump 2  Jump 2  Jump 8  Jump 3  Jump 1  Love! | Place 2 has Valentine's day.  Place 0 has Valentine's day.  Place 0 already had Valentine's day.  Place 0 already had Valentine's day.  Cupid's last position was 1.  Cupid has failed 1 places. |  |

### JS Input / Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| [("10@10@10@2",  "Jump 1",  "Jump 2",  "Love! "]) | Place 3 has Valentine's day.  Cupid's last position was 3.  Cupid has failed 3 places. | Jump 1 ->> [10, 8, 10, 2]  Jump 2 ->> [10, 8, 10, 0] so we print "Place 3 has Valentine's day."  Next command is "Love!", so we print Cupid`s last position and the outcome of his mission. |
| (["2@4@2",  "Jump 2",  "Jump 2",  "Jump 8",  "Jump 3",  "Jump 1",  "Love! "]) | Place 2 has Valentine's day.  Place 0 has Valentine's day.  Place 0 already had Valentine's day.  Place 0 already had Valentine's day.  Cupid's last position was 1.  Cupid has failed 1 places. |  |