**Bonus - Exercises**

**5. Snake Moves**

You are walking in the park and you encounter a snake! You are terrified, and you start running zig-zag, so the snake starts following you.

You have a task to visualize the snake’s path in a square form. A **snake** is represented by **a string**. The **isle** is a **rectangular matrix of size NxM**. A snake starts going down from the **top-left corner** and slithers its way down. The first cell is filled with the first symbol of the snake, the second cell is filled with the second symbol, etc. The snake is as long as it takes in order to **fill the stairs completely** – if you reach the end of the string representing the snake, start again at the beginning. After you fill the matrix with the snake’s path, you should print it.

### Input

* The input data should be read from the console. It consists of exactly two lines
* On the first line, you’ll receive the **dimensions** of the stairs in format: **"N M"**, where **N** is the number of **rows**, and **M** is the number of **columns**. They’ll be separated by a single space
* On the second line you’ll receive the string representing the **snake**

### Output

* The output should be printed on the console. It should consist of **N lines**
* Each line should contain a string representing the respective row of the matrix

### Constraints

* The **dimensions** N and M of the matrix will be integers in the range [1 … 12]
* The **snake** will be a string with length in the range [1 … 20] and **will not contain any whitespace characters**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5 6  SoftUni | SoftUn  UtfoSi  niSoft  foSinU  tUniSo |  |

1. **Jagged Array Manipulator**

Create a program that populates, analyzes and manipulates the elements of a matrix with unequal length of its rows.

First you will receive an **integer N** equal to the **number of rows** in your matrix.

On the **next N lines**, you will receive **sequences of integers**, **separated** by a single **space**. Each sequence is a **row** in the matrix.

After populating the matrix, start analyzing it. If a **row** and the **one below** it have **equal length**, **multiply** each **element** in **both** of them by **2**, **otherwise** - **divide** by **2**.

Then, you will receive commands. There are three possible commands:

* "**Add {row} {column} {value}**" - **add** **{value}** to the element at the **given indexes**, if they are **valid**
* "**Subtract {row} {column} {value}**" - **subtract** **{value}** from the element at the **given indexes**, if they are **valid**
* "**End**" - print the **final state** of the **matrix** (all elements **separated by a single space**) and **stop** the program

**Input**

* On the first line, you will receive the **number of rows** of the matrix - integer **N**
* On the next **N** lines, you will receive **each row** - **sequence of integers**, separated by a single **space**
* **{value}** will always be **integer** number
* Then you will be receiving commands until reading "**End**"

**Output**

* The output should be printed on the console and it should consist of **N lines**
* Each line should contain a string representing the **respective row** of the **final matrix**, elements **separated** by a single **space**

**Constraints**

* The **number of rows** N of the matrix will be integer in the range [2 … 12]
* The **input** will always **follow** the **format above**
* **Think about data types**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  10 20 30  1 2 3  2  2  10 10  End | 20 40 60 1 2 3 2 2 5 5 |
| 5  10 20 30  1 2 3  2  2  10 10  Add 0 10 10  Add 0 0 10  Subtract -3 0 10  Subtract 3 0 10  End | 30 40 60  1 2 3  2  -8  5 5 |