TILING

Let us consider the positive integers k, l, and n. A rectangular room has floor dimensions of $k \cdot n \times l \cdot n$ units (the sign "centered dot" (·) means multiplication of integers). The floor is tiled with $k \cdot l \cdot n$ rectangular tiles of size $1 \times n$. No tile has been cut during the tiling. We can consider the floor as a grid with $k \cdot n$ columns and $l \cdot n$ rows.

A mentally disturbed robot has locked himself in the room and threatens to blow it up if you don't guess the exact way of floor tiling: what is the position of every tile in the covering. Fortunately, the robot is inclined to give you some information, on which basis you can try to reveal the exact tiling. You can ask the robot a set of q questions of the type $(x\ y)$, where $1 \le x \le k \cdot n$ and $1 \le y \le l \cdot n$ are numbers, respectively of a column and a row (counting starts at 1 from the "top left corner" – a cell in first column and first row).

You will get back a set of q ***correct*** answers exactly what tile is covering each of the queried cells: the coordinates of the top left corner ("the beginning") of the tile and the way it is laid (its "direction") – "horizontally" (along a row) or "vertically" (in one column).

Task

Write a function `tiling()`, which will be compiled with a jury's program and will lead a short dialog with the robot to reconstruct the tiling out of the received information.

You are maybe planning to ask a question for each cell, and voilà! Alas, the number q of questions asked should be as small as possible, otherwise you take the risk to annoy the robot, and although your program has guessed the right tiling, an explosion can occur (that is – a zero for the test: see the grading rules).

Implementation details

You should submit to the grading system a file **tiling.cpp**, which contains the function `tiling()`. Your file may contain any other necessary code and names, but should use as a global neither one of the predefined names below, nor the name `main`.

At the beginning of the file there should exist one line with the preprocessor instruction

```
#include "tiling.h"
```

The jury's program declares and implements the following items:

```
struct Data
{int x,y;
 char d;
};
void getArea(int *k, int *l, int *n);
bool getData(int q, Data *quest);
void setResult (const char *res);
```

Description of the defined functions for communication with the robot:

1. Function **`getArea`** will return the room parameters *k*, *l* and *n* (the column count will be *k*·*n*, and the row count will be *l*·*n*).
2. You can call **once** the function **`getData`** with the prototype above. Parameters have the following meaning:
   - *q* is the number of questions;
   - the input/output array of records `quest` contains:
     - before the call – the questions themselves (*x* and *y* are respectively the column and the row of the queried cell). The value of the member named *d* doesn't matter here;
     - after the call – the answers of the robot, namely: *x* and *y* are respectively the column and the row of the beginning of the tile, covering the respective asked cell, and the member *d* has for its value one of these two symbols: `h`, meaning horizontal direction, or `v`, denoting vertical direction. If the function returns **`false`**, that means that either there are incorrect data in the input/output array (say, out of the borders), or the function has already been called. In this case the array will contain zeroes as output values.
3. Before the exit, `tiling()` should call the defined function **`setResult`**, where the symbol array `res` describes the tiling. These are *k*·*l*·*n* symbols (with no delimiter), each being `h` or `v`. Here follows the algorithm for creating `res`:

- Start with no symbols in `res`;
- Consider walking the floor (the grid) by rows from the first to the *l*·*n*-th, and every row by columns from the first to the *k*·*n*-th. If you step on an upper left corner of a tile, you add one symbol to `res`: `h` for horizontal direction of the tile, or `v` for vertical direction. Cells which are not upper left corners (beginnings) are being simply jumped over.

Constraints

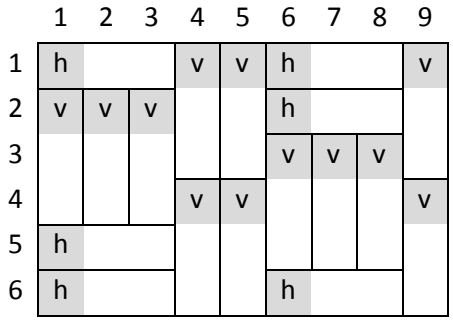
In 20% of the tests, $1 \le k, l \le 10$.
In 30% of the tests, $n = 2$.
$1 \le k, l \le 57$, $2 \le n \le 10$.

Grading

If a registered tiling description is missing or wrong, the test is given no points. A correct description is granted points according to the proximity of the asked questions' count q to the theoretically necessary questions' count. More precisely, if the theoretical minimum is Q questions and the test example is designed for P points, a correct description will be given $\min(P, P(Q/q)^{30})$ points. The final sum is rounded to the nearest integer.

Example

This example is made with respect to the tiling in the picture. You can see in grey the beginnings of the tiles, and a letter in each beginning, denoting the direction of the tile.



If, for example, you have defined variables

```
int k,l,n;
```

a call of the function

```
getArea(&k,&l,&n);
```

will set the values of these variables respectively to `k=3, l=2` and `n=3.`

For an illustration about posing a set of questions, look at the following fragment:

```
Data data[128] = {{1,1,0},{4,1,0},{7,1,0},{1,2,0},
                  {2,3,0},{3,4,0},{4,5,0},{5,6,0},{7,5,0}};
if (getData(9,data)) {//interpret the returned data}
else {//there is an error or this is not the first call}
```

In the considered example after the first call of **`getData`**, as the data before the call are in the border limits, the array `data` will look like this:

```
{{x:1,y:1,d:'h'},{x:4,y:1,d:'v'},{x:6,y:1,d:'h'},
 {x:1,y:2,d:'v'},{x:2,y:2,d:'v'},{x:3,y:2,d:'v'},
 {x:4,y:4,d:'v'},{x:5,y:4,d:'v'},{x:7,y:3,d:'v'}}
```

The result should be registered by calling `setResult`. In this example, say:

```
char r[128]="hvvhvvvvhvvvvvvhhh";
setResult(r);
```

Example comments

Nine correct questions are posed (q=9). The returned nine correct answers allow tiling's determination, which is registered in accordance with the rules. Posed question's count exceeds the theoretical minimum needed for this configuration, which is Q=6. So a solution like this would be granted a part of the provided P points, namely $\{P(2/3)^{30}\}\approx\{0.000005\ P\}$. As you can guess, an answer like this, although correct, will practically be useless.

Local testing

In order to be able to test your function *tiling* on your local computer, you will get files *Lgrader.cpp* and *tiling.h*. Compile *Lgrader* together with your file **tiling.cpp** and you will receive a program that you can use to test your function.

Lgrader reads the standard input in the following format:

- Line 1 contains three integers: k, l and n.
- It follows a matrix of integers. The integer in each cell of this matrix denotes a tile number. The matrix consists of $l \cdot n$ lines; each line contains $k \cdot n$ integers.

Example for local testing (corresponds to the picture above).

Input	Output
3 2 3 1 1 1 2 3 4 4 4 5 6 7 8 2 3 9 9 9 5 6 7 8 2 3 10 11 12 5 6 7 8 13 14 10 11 12 15 16 16 16 13 14 10 11 12 15 17 17 17 13 14 18 18 18 15	hvvhvvvvhvvvvvvhhh