Torusia

Scientists Alison and Bill were on an important scientific expedition for science to Torusia, a newly discovered donut-shaped planet (yes, the cosmologists are still scratching their heads). During some standard experiments, the two scientists were separated and freak solar winds destroyed their communications equipment. They now have no idea how to find each other, however they each have a machine designed for scientific purposes which they can use to re-unite.

Alison and Bill both view Torusia as a 4096 x 4096 grid, with (0, 0) in the **top-left** corner. The grid "wraps around" at the edges so that horizontal lines are in fact horizontal circles and vertical lines are vertical circles. Hence, the point east of (4095, 34) is (0, 34) and the point north of (17, 0) is (17, 4095).

Alison perceives her own position as (0, 0) and Bill perceives his own position as (0, 0) however their absolute positions are different and hence their co-ordinate systems are not aligned. Note however that they have the same orientation (direction of north, south, east, west) so if Alison is x_A metres east and y_A metres south of Bill, then a point Alison refers to as (x, y) would be the point on Bill's grid labelled $((x + x_A) \mod 4096, (y + y_A) \mod 4096)$.

Alison has a machine that performs one function:

- **mark(x, y)**, creates an electromagnetic marker x metres east and y metres south of her position (the cell (x, y) on her grid). Calling **mark(x, y)** on a cell that already contains a marker doesn't create another marker. Alison must ensure that $0 \leq x, y \leq 4095$.

Bill has a machine that can perform two functions:

- **numRow(y)** finds the number of electromagnetic markers that lie on the row y metres south of him. Bill must ensure that $0 \leq y \leq 4095$.
- **numColumn(x)** finds the number of electromagnetic markers that lie on the column x metres east of him. Bill must ensure that $0 \leq x \leq 4095$.

Each minute, Alison and Bill can use their respective machines to perform one function. Alison's machine is a bit faster to start up than Bill's, so each minute you can assume that her marker is placed *before* Bill performs his query.

You are able to send Alison and Bill a program to help Alison place markers and Bill make measurements so that Bill can determine Alison's position as fast as possible.

Library

Your code file will interact with functions provided in the downloadable source file `science.h`. You must implement the following two functions:

- `void alison();` which can make calls to `mark(x, y)`. Note that each time you call `mark`, the current time is increased by one minute. You can call `mark` at most 10 000 times before exiting your function, otherwise the program will terminate execution.
- `void bill();` which can make calls to `numRow(y)` and `numColumn(x)`. Note that each time either of these functions are called, the current time is increased by one minute. Bill must also finally call `found(x, y)` indicating a belief that Alison is x metres east and y metres south of Bill. Calling `found` will terminate execution.

Your code should not have a "main" function, this will be supplied by `science.h`, in addition to the following:

`void mark(int x, int y);` which can only be called (directly or indirectly) by Alison.
`int numRow(int y);` which can only be called (directly or indirectly) by Bill.
`int numColumn(int x);` which can only be called (directly or indirectly) by Bill.
`void found(int x, int y);` which can only be called (directly or indirectly) by Bill.

You may assume that all these functions run in constant time.

Compilation

You must add `#include "science.h"` to the beginning of your code, and make sure the file `science.h` is in the same folder as your code.

Experimentation

Whilst Alison and Bill are in theory simultaneously performing their actions, we are able to simulate this by first running the program "as Alison", which creates a log file of the cell she marks each minute, then running the program "as Bill". The executable created will take one line of standard input that determines which scientist it runs.

- You can enter one line containing the one character '`A`', which will run Alison's code and output to a file `mark_log`. Pro-tip: this file can be useful for debugging!
- You can enter the character '`B`' followed by two space-separated integers x_a and y_a, representing Alison's position relative to Bill. This will run Bill's code and output the result. You must run Alison before running Bill, as Bill will require the file `mark_log` to be present.

You may wish to temporarily change the value of the constant `SIZE` at the top of `science.h`, to test your program on smaller grids.

Sample Session

The following sample session is based on this code:

```
#include "science.h"

void alison() {
    mark(1, 100);
    mark(2000, 100);
}

void bill() {
    int a = numRow(120);
    int b = numColumn(904);
    int c = numRow(120);
    found(3000, 20);
}
```

First, the program is run and the following input is provided to standard input (the screen):

```
A
```

The program will run `alison()`, which makes these function calls:

Function Call	Explanation
`mark(1, 100)`	In the first minute, Alison marks the cell 1m east and 100m south of her.
`mark(2000, 100)`	In the second minute, Alison marks the cell 2000m east and 100m south of her.

The program exits successfully and creates a file `mark_log`. We now run the program again and provide the following input, which places Alison 3000m east and 20m south of Bill.

```
B 3000 20
```

The program will run `bill()`, which makes these function calls:

Function Call	Explanation
`numRow(120)`	returns 1, as in the first minute Alison marked a cell 100m south of her, and she is 20m south of Bill, hence there is one marked cell on the row 120m south of Bill.
`numColumn(904)`	returns 1, as in the second minute Alison, 3000m east of Bill, marked a cell 2000m east, which is the cell only 904m east of Bill since the grid is 4096m wide.
`numRow(120)`	returns 2, as there are now two markers on the row 120m south of Bill
`found(3000, 20)`	Through some amazingly lucky guesswork, Bill correctly finds Alison's position after only 3 minutes.

Scoring

Your program will be given a score based on the time it takes Bill to find Alison. Specifically, if `found` is called with the correct parameters after M minutes:

$$\text{score} = \begin{cases} 100, & \text{if } M \leq 144 \\ \frac{13000}{M} + 10, & \text{if } 144 < M \leq 10\,000 \\ 0, & \text{if } 10\,000 < M \end{cases}$$

Here is a table showing ten example scores and corresponding success time required:

Score	11	20	30	40	50	60	70	80	90	100
M	10 000	1 300	650	433	325	260	216	185	162	144

There are no subtasks for this problem. Your total score will be the minimum score your program received over all test cases.