Bombs

Michael the Explorer, along with his friend Nathan, are on a special mission to destroy an evil spy network that has been operating extensively in the world today. Their base is located in the distant land of Azaria. Their leader, Jonathan, is their main target. This is a special, top-secret mission that is strictly confidential. It is unknown how the problem setter somehow receives information about their top secret mission.

The spies, known as "targets", are based in a rectangular grid of size R-by-C, with R rows and C columns. They need to hit the targets using the bombs they carry inside their aircraft. These bombs have different explosion range and thus will destroy an area corresponding to the range of each bomb.

Michael can choose to drop the bomb in any cell inside the grid. The grid, however, has a magical wall surrounding it that can nullify explosions. Therefore, if the bomb somehow has an explosion range outside the grid (even though it is dropped inside), the explosion on those outside cells are ignored. Therefore, the civilians living outside of the grid will be left unharmed.

All the bombs have an odd-size explosion range. If a bomb with an explosion range of N is dropped at cell (x, y), then the cells that form a square with $\left(x - \frac{N-1}{2}, y - \frac{N-1}{2}\right)$ as the top-left corner and $\left(x + \frac{N-1}{2}, y + \frac{N-1}{2}\right)$ with the bottom-right corner will be destroyed.

Michael has now started his mission and discover the following grid: (the circles denote the targets)

O				
	O	O	O	
	O		O	
			O	

He decides to drop a size-3 bomb in the middle cell (2,2), destroying 6 targets:

O				
	O	O	O	
	O		O	
			O	

The darker cell indicates the cell where the bomb was deployed, and the lighter ones indicate the exploded area. (Note that we use 0-based indexing).

Having destroyed many targets, Michael is satisfied. He then decides to fly back home. However, to his surprise, there are still many grids left in the battlefield! Some of these are so huge that he himself

cannot devise an efficient strategy to destroy the targets. Fortunately, his friend, Nathan is an excellent programmer. Michael will ask him two types of queries, and he needs to give a fast answer to Michael using his program. The queries are:

1. You have a bomb of size N, which is guaranteed to be an odd-number. Given this bomb with size N, in which cell does Michael need to deploy it such that it destroys the most number of targets? There can be more than one query of this type.

2. You want to destroy the most number of targets but want to minimize the resources used. Therefore, what bomb size and where should it be deployed, such that we minimize the bomb size and kill the targets efficiently and effectively? This query will only be asked once, and after all of the previous queries (of the first type) have been asked.

For the second query type, Michael has defined a point system to determine the efficiency and effectiveness of the explosion. For each target hit, 3 (three) points are received. However, for each empty cell hit, 1 (one) point is deducted. Therefore, the most effective and efficient answer is the one yielding the most number of points. For example, the bomb deployed in the grid above will yield 15 points since it hit 6 targets with 3 empty cells being hit. It is the most efficient and effective drop for the above grid.

Before starting, there are a few guidelines that the program needs to adhere to:

1. Bomb sizes must be an odd number, with the smallest size being 1 (one).
2. There are no empty grids.
3. The bombs can be deployed in any cell inside the grid.
4. For query type 1, if there are multiple locations where the bomb will destroy the most number of targets, pick the one with the smallest row number first, followed by the smallest column number if there is still a tie. update if bigger
5. For query type 2, if there is a tie, we first want to minimize the bomb size, followed by the deployment position similar to the condition for query type 1.

This is an important mission. Make your country proud by implementing this sophisticated program that can help Michael accomplish his mission.

Input

The first line of input consists of two numbers, **R** and **C** (3 <= **R, C** <= 50), separated by a single space, denoting the number of rows and columns of the grid respectively. The next R lines will have C characters, separated by a single space, each denoting whether a cell contains a target. A target is denoted by “1” and “0” denotes an empty cell.

The next line of input is a single integer **Q** (**Q** <= 5), the number of type-1 queries. The next Q lines will contain a single integer N. N is the size of the bomb being asked as part of the first query type.

Output

For the first Q lines of output, print two numbers, separated by a single space, the x and y-coordinates of the deployment position which satisfy the first query type. Note that we use 0-based indexing.

The last line of the output consists of three numbers, separated by a single space, denoting the x-coordinate, y-coordinate, and the bomb size which satisfies the second query type for the given grid.

Your last line of output should contain a newline character.

Sample Input

```
5 5
1 0 0 0 0
0 1 1 1 0
0 1 0 1 0
0 0 0 1 0
0 0 0 0 0
1
3
```

Sample Output

```
2 2
2 2 3
```

Explanation

The sample input is the map used in the problem description. There is only 1 type-one query being asked. The bomb with size 3 is best deployed in cell (2,2). For the second query, we also need to deploy a size-3 bomb in the cell (2,2) to get the best solution. We can use a size-1 bomb in cell (0,0) which will hit 1 target and no empty cells, but it will only yield us 1 point. We can also destroy the entire grid by dropping a size-5 bomb in the cell (2,2). However, the points yield is only 3 points because of the penalty.

Clarifications

If we drop a size-3 bomb in cell (0,0) based on the map below, the explosion will look like this:

O		
	O	O
	O	

Skeleton

You are given the skeleton file Bombs.java.

Notes:

1. You should develop your program in the subdirectory ex1 and use the skeleton java file provided. You **should not create a new file or rename the file provided.**
2. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm.
3. You do not have to use OOP for this sit-in lab. However, you are reminded to pay attention to the **modularity** of your program.
4. You are free to define your own helper methods.
5. Please be reminded that the marking scheme is:

Input	: 10%	**Correctness**	: 50%
Output	: 10%	**Programming Style**	: 30%