Problem 4

Bomberman!

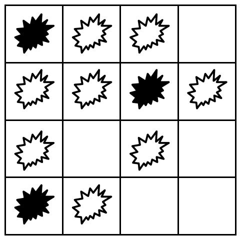
6 Points

Program Name: bomberman.cpp   Input File: bomberman.dat

Introduction

You are a huge Bomberman fan, but the only problem is that you aren't necessarily that good at it. You are particularly bad at the levels where you have to avoid raining bombs from the heavens. You have played each level so many times that you know exactly where the bombs are going to drop, but your only problem is that it seems no matter what sequence of moves you perform, you always seem to get clobbered! You have come up with the idea that if you could figure out how to find a spot that is never bombed, then you could go there and not have to move at all and pass the level! Luckily your programming skills are much better than your Bomberman skills, so you quickly whip up a program to calculate the safe spots given the bomb trajectories.

The area map that Bomberman is in can be broken up into a grid of squares. Bomberman can occupy one square and the trajectory of a bomb is the series of squares in which the bomb affects. This includes the square in which the bomb lands, along with one square in each adjacent direction (up, down, left, and right). Bombs do not affect diagonal squares, and can only affect squares that are located on the area map. If Bomberman occupies one of these affected squares, he is toast! Here is an example of a 4x4 area map where the coordinates of each square go from 0-3 (starting from the top left-hand side of the area map), and the locations of the resulting trajectories of three bombs given they were dropped on squares (0,0), (1,2), (3,0):



The safe spots for Bomberman would then be squares (0,3), (2,1), (2, 3), (3,2), and (3,3). In other words, if Bomberman would stay in any of these spots, he could avoid the barrage of bombs without moving, hooray!

Input

Input to this problem will consist of a (non-empty) series of up to 100 data sets. Each data set will be formatted according to the following description, and there will be **no blank lines** separating data sets.

A single data set has 3 components:

1. *Start line* - A single line, "START X Y Z", where X is number of rows in the area map, Y is the number of columns in the area map, and Z is the number of bombs that will be dropped; where (1 <= X <= 10), (1 <= Y <= 10), and (1 <= Z <= 5).
2. *Bomb Positions* - A single line consisting of Z positions on the area map that a bomb will be dropped. Each bomb position will be of the format (Bx,By), where 'Bx' is the row coordinate and 'By' is the column coordinate that the bomb will be dropped in. If there is more than one bomb position, each bomb position will be separated by a space.
3. *End line* - A single line, "END"

Note: *All bomb position coordinates start at 0.*

Output

For each data set, there will be exactly one line of output. Each line of output will consist of a list of coordinates that are not affected by any of the dropped bombs, and are thus safe spots for the Bomberman to stand. Each safe position will be printed out in the format (Sx,Sy), where 'Sx' is the row coordinate and 'Sy' is the column coordinate of the safe position. The safe coordinates shall be printed in ascending order starting with Sx, then followed by Sy. If there is more than one safe position for a given data set, they will be separated by a single space. If no safe positions exist, the string "BOMBERMAN'S TOAST!" will be printed instead.

Note: *All safe position coordinates start at 0.*

Example: Input File

```
START 4 4 3
(0,0) (1,2) (3,0)
END
START 5 3 5
(0,0) (0,2) (2,1) (4,0) (4,2)
END
```

Output to screen

```
(0,3) (2,1) (2, 3) (3,2) (3,3)
BOMBERMAN'S TOAST!
```