Problem G – N-Credible Mazes

Background

An *n-tersection* is defined as a location in *n*-dimensional space, *n* being a positive integer, having all non-negative integer coordinates. For example, the location (1,2,3) represents an n-tersection in three dimensional space. Two n-tersections are said to be *adjacent* if they have the same number of dimensions and their coordinates differ by exactly 1 in a single dimension only. For example, (1,2,3) is adjacent to (0,2,3) and (2,2,3) and (1,2,4), but not to (2,3,3) or (3,2,3) or (1,2). An *n-teresting space* is defined as a collection of paths between adjacent n-tersections. Finally, an *n-credible maze* is defined as an n-teresting space combined with two specific n-tersections in that space, one of which is identified as the starting n-tersection and the other as the ending n-tersection.

Input

The input file will consist of the descriptions of one or more n-credible mazes. The first line of the description will specify *n*, the dimension of the n-teresting space. (For this problem, *n* will not exceed 10, and all coordinate values will be less than 10.) The next line will contain 2*n* non-negative integers, the first *n* of which describe the starting n-tersection, least dimension first, and the next *n* of which describe the ending n-tersection. Next will be a non-negative number of lines containing 2*n* non-negative integers each, identifying paths between adjacent n-tersections in the n-teresting space. The list is terminated by a line containing only the value –1. Several such maze descriptions may be present in the file. The end of the input is signalled by space dimension of zero. No further data will follow this terminating zero.

Output

For each maze output it's position in the input; e.g. the first maze is "Maze #1", the second is "Maze #2", etc. If it is possible to travel through the n-credible maze's n-teresting space from the starting n-tersection to the ending n-tersection, also output "can be travelled" on the same line. If such travel is not possible, output "cannot be travelled" instead.

Example

Input

```
2
0 0 2 2
0 0 0 1
0 1 0 2
0 2 1 2
1 2 2 2
-1
3
1 1 1 1 2 3
1 1 2 1 1 3
1 1 3 1 2 3
1 1 1 1 1 0
1 1 0 1 0 0
1 0 0 0 0 0
-1
0
```

Output

```
Maze #1 can be travelled
Maze #2 cannot be travelled
```