The following conventions must be followed for computing operational path using each algorithm:

1. **BFS:**
   In BFS, each move from one cell to any of its 8 neighbors counts for a unit path cost of 1. You do not need to worry about elevation differences (except that you still need to ensure that they are allowable and not too steep for your rover), or about the fact that moving diagonally (e.g., North-East) actually is a bit longer than moving along the North to South or East to West directions. So, any allowed move from one cell to an adjacent cell cost 1.
2. **UCS:**
   When running UCS, you should compute unit path costs in 2D. Assume that cells' center coordinates projected to the 2D ground plane are spaced by a 2D distance of 10 North-South and East-West. That is, a North or South or East or West move from a cell to one of its 4-connected neighbors incurs a unit path cost of 10, while a diagonal move to a neighbor incurs a unit path cost of 14 as an approximation to $10\sqrt{2}$ when running UCS.
3. **A* search:**
   When running A*, you should compute an approximate integer unit path cost of each move in 3D, by summing the horizontal move distance as in the UCS case (unit cost of 10 when moving North to South or East to West, and unit cost of 14 when moving diagonally), plus the absolute difference in elevation between the two cells. For example, moving diagonally from one cell with Z=20 to adjacent North-East cell with elevation Z=18 would cost 14+|20-18|=16. Moving from a cell with Z=-23 to adjacent cell to the West with Z=-30 would cost 10+|-23+30|=17. An admissible heuristic is to be designed for this algorithm.

Input: The file input.txt in the current directory of your program will be formatted as follows:

First line: Instruction of which algorithm to use, as a string: BFS, UCS or A*

Second line: Two strictly positive 32-bit integers separated by one space character, for "W H" the number of columns (width) and rows (height), in cells, of the map.

Third line: Two positive 32-bit integers separated by one space character, for "X Y" the coordinates (in cells) of the landing site. 0 <= X <= W-1 and 0 <= Y <= H-1 (that is, we use 0-based indexing into the map; X increases when moving East and Y increases when moving South; (0,0) is the North West corner of the map).

Fourth line: Positive 32-bit integer number for the maximum difference in elevation between two adjacent cells which the rover can drive over. The difference in Z between two adjacent cells must be smaller than or equal (<=) to this value for the rover to be able to travel from one cell to the other.

Fifth line: Strictly positive 32-bit integer N, the number of target sites.

Next N lines: Two positive 32-bit integers separated by one space character, for "X Y" the coordinates (in cells) of each target site. 0 <= X <= W-1 and 0 <= Y <= H-1 (that is, we again use 0-based indexing into the map). Next H lines: W 32-bit integer numbers separated by any numbers of spaces for the elevation (Z) values of each of the W cells in each row of the map.

For example:

A*

8 6

4 4

7

2

1 1

6 3

0 0 0 0 0 0 0 0

0 60 64 57 45 66 68 0

0 63 64 57 45 67 68 0

0 58 64 57 45 68 67 0

0 60 61 67 65 66 69 0

0 0 0 0 0 0 0 0

In this example, on 8-cells-wide by 6-cells-high grid, we land at location (4, 4) highlighted in green above, where (0, 0) is the North-West corner of the map. The maximum elevation changes that the rover can handle is 7 (in arbitrary units which are the same as for the Z values of the map). We want to visit 2 targets, at locations (1, 1) and (6, 3), both highlighted in red above. The Z elevation map is then given as six lines in the file, with eight Z values in each line, separated by spaces.

Output: The file output.txt which your program creates in the current directory should be formatted as follows:

N lines: Report the paths in the same order as the targets were given in the input.txt file. Write out one line per target. Each line should contain a sequence of X, Y pairs of coordinates of cells visited by the rover to travel from the landing site to the corresponding target site for that line. Only use a single comma and no space to separate X, Y and a single space to separate successive X, Y entries. If no solution was found (target site unreachable by rover from given landing site), write a single word FAIL in the corresponding line.

For example, output.txt may contain:

4,4 3,4 2,3 2,2 1,1

4,4 5,4 6,3

The first path looks like this:

0 0 0 0 0 0 0 0

0 60 64 57 45 66 68 0

0 63 64 57 45 67 68 0

0 58 64 57 45 68 67 0

0 60 61 67 65 66 69 0

0 0 0 0 0 0 0 0

With the landing site shown in green, the target site in red, and each traversed cell in between in yellow. Note how one could have thought of a perhaps shorter path: 4,4 3,3 2,2 1,1 (straight diagonal from landing site to target site). But this was not possible for this rover as the move from 4,4 to 3,3 would incur a difference in Z of |65 – 57| = 8 which is too steep for this rover (difference must be <= 7 according to input.txt for this example).

And the second path looks like this:

0	0	0	0	0	0	0	0
0	60	64	57	45	66	68	0
0	63	64	57	45	67	68	0
0	58	64	57	45	68	67	0
0	60	61	67	65	66	69	0
0	0	0	0	0	0	0	0