H• Histology Assistant

Problem

An application to assist in the analysis of tissue samples is to work as follows. A digital microphotograph of a stained tissue sample is scanned to identify stained pixels. For each region of stained pixels, an outline of the region is obtained. The outline is then analyzed for shape indicators of disease and the outlines (color-coded for possible disease) are overlaid on the microphotograph as it is displayed to the pathologist.

This problem is to write a program, which processes a bitmap of stained and unstained pixels, finds regions of stained pixels and, for each region, outputs the outline of the region. Regions with fewer stained pixels than a *minimum size* are ignored. Only the outer boundary is computed (interior holes are ignored).

A pixel is *adjacent* to another pixel if the second pixel is directly above, directly below, directly left or directly right of the first pixel. Two stained pixels are *connected* if there is a sequence of stained pixels starting with one of the pixels and ending with the other for which each pixel in the sequence is adjacent to the next. A *region* of stained pixels is a set of stained pixels, all of which are connected to a single stained pixel. A stained pixel is a *boundary pixel* of its region if at least one of the pixels adjacent to it is not stained. (All pixels immediately outside the bitmap are considered unstained so that pixels on the edge of the bitmap are boundary pixels.) In the example below, there are 4 regions (stained pixels are 'X', unstained are '`.`').

```
........................................
.XX.....................................
..X.................XXX......XXX........
.....................XXX....XXX.........
.......XXX............XXX..XXX..........
.....XXXXXXX...........XXXXXX...........
....XXXXXXXXXX..........XXXX............
...XXXX...XXXX...........XX.............
..XXX.......XXX.........................
..XXX.......XXX........XXXXXXX..........
.XXX.........XXX.......XXXXXXX..........
.XXX.........XXX.......XXXXXXX..........
.XXX.........XXX.......XXXXXXX..........
..XXX.......XXX...........X.............
..XXX.......XXX...........X.............
...XXXX...XXXX.........XXXXXXX..........
....XXXXXXXXXX.........XXXXXXX..........
.....XXXXXXX...........XXXXXXX..........
.......XXX.............XXXXXXX..........
........................................
```

Outlines are to be specified as the left most point of the top most line of the region, a count of boundary pixels and a sequence of moves from one boundary pixel to the next clockwise using the codes (*up* = A, *up right* = B, etc.):

```
H  A  B
G     C
F  E  D
```

Rows are numbered from top to bottom beginning with 1. Columns are numbered from left to right beginning with 1. For example the outline of the 'v' shaped region above would be:

3 21 22
CCDDDCBBBCCFFFFFGHHHHH

Input

Input is a sequence of *problem instances*. Each *problem instance* begins with a line containing 3 decimal numbers: *row-count*, *column-count* and *minimum-number-of-pixels*. This line is followed by *row-count* lines of *column-count* characters. Each character is either a period (**.**) for an unstained pixel or an upper-case X for a stained pixel. The input ends when the *row-count* is 0. *Row-count* will be at most 47, *column-count* will be at most 63 and *minimum-number-of-pixels* will be at least 2.

Output

For each *problem instance*, the output begins with a line starting with a decimal integer giving the *number of components* of at least *minimum-number-of-pixels* stained pixels. This is followed by the description of the boundary of each component. The boundaries are to be listed in the order that a first pixel of the component appears while scanning across lines from left to right with line scanned from top to bottom. For each component, the output begins with a line giving the *row number* of the start pixel, the *column number* of the start pixel and the number of pixels in the boundary as decimal integers separated by a single space. This line is followed by lines of direction codes 'A' through 'H'. Each line shall have 40 characters except the last line.

Example

Input

```
20 40 4
........................................
.XX.....................................
..X.................XXX......XXX........
.....................XXX....XXX.........
.......XXX............XXX..XXX..........
.....XXXXXXX...........XXXXXX...........
....XXXXXXXXX...........XXXX............
...XXXX...XXXX...........XX.............
..XXX.......XXX.........................
..XXX.......XXX........XXXXXXX..........
.XXX.........XXX.......XXXXXXX..........
.XXX.........XXX.......XXXXXXX..........
.XXX.........XXX.......XXXXXXX..........
..XXX.......XXX...........X.............
..XXX.......XXX...........X.............
...XXXX...XXXX.........XXXXXXX..........
....XXXXXXXXX..........XXXXXXX..........
.....XXXXXXX...........XXXXXXX..........
.......XXX.............XXXXXXX..........
........................................
12 40 4
.X.X.X.X.X.X......XX...XXXXXXXXXXXXXXXX.
.XXX.XXX.XXX......XX..XXXXXXXXXXXXXXXXXX
.X...X...X........XX..XX.............XXX
.X.X.X.X.X.X......XX..XX...XXXXXXXX...XX
.XXX.XXX.XXX......XX..XX..XXXXXXXXXX..XX
.X...X...X........XX..XX..XX......XX..XX
.X.X.X.X.X.X......XX..XXX........XXX..XX
.XXX.XXX.XXX......XX..XXXXXXXXXXXXXX..XX
.X...X...X........XX...XXXXXXXXXXXX...XX
.X.X.X.X.X.X......XXX................XXX
.XXXXXXXXXXX......XXXXXXXXXXXXXXXXXXXXXX
...................XXXXXXXXXXXXXXXXXXXX.
0 0 0
```

Output

```
3
3 21 22
CCDDDCBBBCCFFFFFGHHHHH
5 8 36
CCDCDDDEDEEFEFFFGFGGHGHHHAHAABABBBCB
10 24 38
CCCCCCEEEGGFEDCCEEEGGGGGGAAACCBAHGGAAA
2
1 2 103
DBEGFEDBEGFEDBEGFEDBDBAAAAAAAAADBEGFEDBE
GFEDBEGFEDBDBAAAAAAAAADBEGFEDBEGFEDBEGFE
DBEGGGGGGGGGGAAAAAAAAAA
1 19 159
CEEEEEEEEDDCCCCCCCCCCCCCBBAAAAAAAHHGGGGG
GGGGGGGFEEEDDCCCCCCCBBHGGGGGFGABCCCCCCCD
EEEFGGGGGGGGGGGHAAAAAABCCCCCCCCCCCCCCCDE
EEEEEEEEFGGGGGGGGGGGGGGGGGGHAAAAAAAAAAA
```