You are working for an amusement park as an operator of an *obakeyashiki*, or a haunted house, in which guests walk through narrow and dark corridors. The house is proud of their lively ghosts, which are actually robots remotely controlled by the operator, hiding here and there in the corridors. One morning, you found that the ghosts are not in the positions where they are supposed to be. Ah, yesterday was Halloween. Believe or not, paranormal spirits have moved them around the corridors in the night. You have to move them into their right positions before guests come. Your manager is eager to know how long it takes to restore the ghosts.

In this problem, you are asked to write a program that, given a floor map of a house, finds the smallest number of steps to move all ghosts to the positions where they are supposed to be.

A floor consists of a matrix of square cells. A cell is either a wall cell where ghosts cannot move into or a corridor cell where they can.

At each step, you can move any number of ghosts simultaneously. Every ghost can either stay in the current cell, or move to one of the corridor cells in its 4-neighborhood (i.e. immediately left, right, up or down), if the ghosts satisfy the following conditions:

1. No more than one ghost occupies one position at the end of the step.
2. No pair of ghosts exchange their positions one another in the step.

For example, suppose ghosts are located as shown in the following (partial) map, where a sharp sign ('#') represents a wall cell and 'a', 'b', and 'c' ghosts.

```
####
 ab#
#c##
####
```

The following four maps show the only possible positions of the ghosts after one step.

```
####   ####   ####   ####
 ab#   a b#   acb#   ab #
#c##   #c##   # ##   #c##
####   ####   ####   ####
```

Input

The input consists of at most 10 datasets, each of which represents a floor map of a house. The format of a dataset is as follows.

$$\begin{array}{llll} w & h & n & \\ c_{11} & c_{12} & \cdots & c_{1w} \\ c_{21} & c_{22} & \cdots & c_{2w} \\ \vdots & \vdots & \ddots & \vdots \\ c_{h1} & c_{h2} & \cdots & c_{hw} \end{array}$$

w, h and n in the first line are integers, separated by a space. w and h are the floor width and height of the house, respectively. n is the number of ghosts. They satisfy the following constraints.

$$4 \leq w \leq 16, \quad 4 \leq h \leq 16, \quad 1 \leq n \leq 3$$

Subsequent h lines of w characters are the floor map. Each of c_{ij} is either:

- a '#' representing a wall cell,
- a lowercase letter representing a corridor cell which is the initial position of a ghost,
- an uppercase letter representing a corridor cell which is the position where the ghost corresponding to its lowercase letter is supposed to be, or
- a space representing a corridor cell that is none of the above.

In each map, each of the first n letters from a and the first n letters from A appears once and only once. Outermost cells of a map are walls; i.e. all characters of the first and last lines are sharps; and the first and last characters on each line are also sharps. All corridor cells in a map are connected; i.e. given a corridor cell, you can reach any other corridor cell by following corridor cells in the 4-neighborhoods. Similarly, all wall cells are connected. Any 2×2 area on any map has at least one sharp. You can assume that every map has a sequence of moves of ghosts that restores all ghosts to the positions where they are supposed to be.

The last dataset is followed by a line containing three zeros separated by a space.

Output

For each dataset in the input, one line containing the smallest number of steps to restore ghosts into the positions where they are supposed to be should be output. An output line should not contain extra characters such as spaces.

Sample Input

```
5 5 2
#####
#A#B#
#   #
#b#a#
#####
16 4 3
################
## ########## ##
#    ABCcba    #
################
16 16 3
################
### ##    #   ##
##  #  ##   # c#
#  ## ########b#
# ##  # #   #  #
#  # ##   # # ##
##  a#  # # #  #
### ## #### ## #
##   #   #  #  #
#  ##### # ## ##
####   #B# #   #
##  C#   #   ###
#  # # ####### #
# ######  A##  #
#        #    ##
################
0 0 0
```

Sample Output

```
7
36
77
```