Arrangement

Input file:	`input_*.txt`
Output file:	`output_*.txt`
Time limit:	N/A
Memory limit:	N/A
Points available:	100

Ravi has just built a number of new restaurants and requires your assistance to fill them with tables as optimally as possible.

Each restaurant is modelled as an $N \times M$ grid of 1×1 cells, and each cell is either empty, blocked (by a wall), or is a "door". There is exactly one door for each restaurant. A table type consists of a number of 1×1 cells joined along their edges. Only a limited variety of table types are available for a particular restaurant, but you can use an available table type as many times as you like.

To place a table of a specific type in the restaurant (at some offset), the corresponding cells in the restaurant must all be empty. Upon placing the table, these cells of the restaurant become blocked. Tables can only be placed in the orientation given (they are patterned to match the restaurant's orientation). You are required to place tables into the restaurant such that for each placed table, either the table is adjacent to the door, or there is a path whose first cell starts adjacent to the door and whose last cell is adjacent to any cell of that table.

For each restaurant, Ravi also has in mind a target number of cells to be covered by tables, K, which he will use to grade your solution. **If Ravi receives a solution (from any contestant) which beats his target, the value of K will be increased to match it**.

Two cells are *adjacent* if they share an edge.

A *path* is a list of one or more empty cell coordinates, such that every two consecutive cells in the list are adjacent.

Input

NB: Attachments for this question are found on the contest website, in the "Statement" tab of this question. You will see a section **Attachments** with "tests.zip" and "tables.txt".

The different table types are recorded in "tables.txt" (attached), where each table type is represented by a grid of empty and blocked cells, with the blocked cells showing the shape of the table.

The first line of this file indicates the number of different table types.

Then, for each table, there is a line consisting of 3 space-separated integers, a, b and c, meaning the table is type a, and a further b lines follow containing c characters each, the grid describing the table. A '`#`' character indicates the cell is part of the table, while a '`.`' character indicates an empty cell.

This is an output-only task, so the input files ("input_1.txt", ..., "input_8.txt") are attached in "tests.zip", and must be downloaded. You are able to view the exact input, and tailor your programs as needed.

Each input file corresponds to a single restaurant.

The first line contains four integers separated by spaces: N, M, C and K.

The next line contains C integers separated by spaces, indicating the table types available for the restaurant.

The next N lines contain M characters each, describing the restaurant. A '`.`' indicates an empty cell, a

'#' indicates a blocked cell, and a 'D' indicates the door. Each restaurant is bordered by walls, with the door found on the left border.

Output

You must submit only the corresponding output file produced for each input file. The output file for "input_*.txt" must be named "output_*.txt", where the "*" indicates the file number.

In the first line of each output file, output a single integer T, the number of tables to be placed.

The next T lines describe how to place the tables. Each must consist of 3 integers separated by spaces. The first integer must indicate the type of table, and the next two integers must be the vertical and horizontal offset at which to place the table.

The vertical offset is the vertical distance between the top edge of the restaurant and the top edge of the table, and the horizontal offset is the horizontal distance between the left edge of the restaurant and the left edge of the table.

Scoring

There is one input file for each restaurant, with 12.5 marks allocated to each.

For each restaurant, if your solution does not satisfy the constraints, you will score 0; however, if there is no path to a placed table once all tables are placed, such a table will instead just be ignored, and you will score points for the remaining tables. If your solution results in L cells being covered by tables, you will score

$$40 \cdot \frac{L}{K} + 40 \cdot \left(\frac{L}{K}\right)^2 + 20 \cdot \max\left(0,\ 10 \cdot \frac{L}{K} - 9\right)^2$$

percent of the marks available.

A solution scoring full marks is guaranteed to exist for each restaurant.

Examples

input_*.txt	output_*.txt
5 5 2 5 1 4 ##### D..## #...# #...# #####	2 1 1 2 4 2 1
5 6 3 3 1 3 7 ###### ##.#.# D..#.# ##.#.# ######	4 7 1 2 1 1 4 1 2 4 1 3 4

Note

In the first example output, a type 1 table is placed in the top middle of the room, while a type 4 table is placed in the bottom left corner. After tables have been placed, the room would look like:

```
#####
D.A##
#.B.#
#BB.#
#####
```

(where `A` indicates the type 1 table and `B` indicates the type 4 table).

A total of 4 cells were covered by tables; hence, this solution would score

$$40 \cdot \frac{4}{5} + 40 \cdot \left(\frac{4}{5}\right)^2 + 20 \cdot \max\left(0,\ 10 \cdot \frac{4}{5} - 9\right)^2 \ = \ 32 + 25.6 + 0 \ = \ 57.6$$

percent of the marks available.

In the second example output, after tables have been placed, the room would look like:

```
######
##C#E#
D.C#F#
##C#G#
######
```

(where `C` indicates the type 7 table and `E`, `F` and `G` indicate the three type 1 tables)

The type 1 tables can not be reached, so they are ignored. The type 7 table is included as it can be reached from the door. It covers 3 cells, so this solution would score 100%.