Czech ACM Student Chapter
Charles University in Prague
University of West Bohemia

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina

CTU Open Contest 2021

Tone Banks

`tone.c`, `tone.cpp`, `Tone.java`, `tone.py`

On Mars, the greatest Earth music hits of all time are stored in huge physical databases of records. This will allow current and future Mars inhabitants to remind themselves of the culture of the motherland, should they grow homesick. These databases are called Tone Banks, as they store the songs tone by tone.

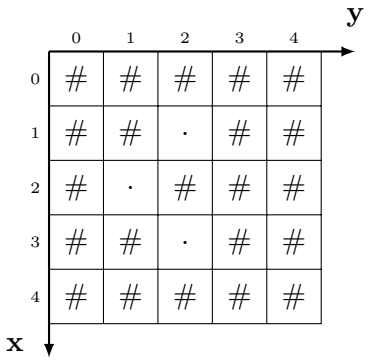
However, only just recently, it has been discovered the data format of the records might degrade when exposed to cosmic radiation over long periods of time. Because of that, is it needed to quickly convert the records from the currently used format to a new one.

The current data format consists of a 2D grid of Data Points. Each Data Point is of one of two types, either a "`#`" or a "`.`" (similarly to Earth's 1s and 0s). These two Data Point types are said to be complementary to each other. Two Data Points are considered adjacent, if they share an edge in the grid. This means a Data Point on coordinates (X, Y), that is, on X-th row and Y-th column, is adjacent to Data Points on coordinates $(X + 1, Y)$, $(X - 1, Y)$, $(X, Y + 1)$ and $(X, Y - 1)$ (within the grid range).

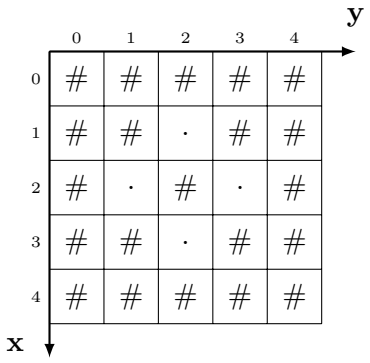
Data Points form so called Data Blobs, which are connected areas consisting of a particular Data Point type. That means each Data Point of a Data Blob is reachable from every other Data Point of the Data Blob via a sequence of steps through consecutively adjacent Data Points of the same type.

On top of that, within the connected area of a Data Blob, there may be other Data Blobs consisting of the complementary Data Point type. The containment may repeat indefinitely, alternating Data Point types, until there is a Data Blob not containing any other Data Blob. For a Data Blob, we call inner Data Blobs such blobs within its area, that are adjacent to at least one Data Point of the Data Blob. Each Data Blob may contain 0 to 26 inner Data Blobs. The number of inner Data Blobs contained in a Data Blob determine which tone, represented as an English character, it encodes (1 Data Blob = a, 2 Data Blobs = b, ..., 26 Data Blobs = z). A Data Blob not containing any other Data Blobs doesn't represent any tone/character; it represents an empty string. A Data Blob must encompass its inner Data Blobs fully within itself. That means each Data Point (X, Y) of its inner Data Blobs must not be adjacent with Data Points of the same type of its, possibly hypothetical, outer Data Blob (the Data Blob in which it is contained). These points must not only not be adjacent according to the definition above, but they must not be adjacent even over a diagonal to points $(X+1, Y+1)$, $(X+1, Y-1)$, $(X - 1, Y + 1)$ and $(X - 1, Y - 1)$. Note that different inner Data Blobs of a single Data Blob may be adjacent over a diagonal. See images below for examples.

Each stored song consists of a sequence of tones, and thus it may be represented as a word consisting of English letters. A song is decoded from a record by the following algorithm. For simplicity we consider that beyond the boundaries of the grid there is an infinite Data Blob of type "`.`". In this Data Blob, there is always a single Data Blob of type "`#`". We start by decoding this Data Blob. The word represented by a Data Blob is a concatenation of the letter representing the Data Blob and all words represented by inner Data Blobs contained within it. The concatenation happens consecutively by the lexicographical ordering of the coordinates



3 correct inner "." Data Blobs



Incorrect inner "#" Data Blob

(X, Y) of Data Points of the inner Data Blobs. That is, from two Data Blobs, we first concatenate the word representing a Data Blob with a Data Point with the least row number, or in the case of equality of row numbers, the least column number.

The new data format adheres to the very same rules, only it encodes the reverse of the word encoded by the old format. Can you safely transform all the songs to the new format?

Input Specification

On the input there are two integers N and M ($1 \leq N, M \leq 100$), representing the number of rows and the number of columns in the grid of the old data format, respectively. Next N lines follow, each containing M characters of type either "`#`" or "`.`". The grid encodes a nonempty word in accordance to the format described above.

Output Specification

Output two integers K and L ($1 \leq K, L \leq 3000$), the number of rows and the number of columns of the grid for the new data format, respectively. After that, output K lines with L characters each, of type either "`#`" or "`.`". The grid must encode the reverse of the word encoded by the input.

Sample Input 1

```
7 7
#######
#.....#
#.....#
#.....#
#.....#
#.....#
#######
```

Output for Sample Input 1

```
3 3
###
#.#
###
```

The first Sample Input encodes word "a", therefore the answer encodes also word "a".

Sample Input 2

```
8 29
#############################
#...##........##............#
#.#.##.#..##..##.####...###.#
#.#.##....##..##.####...#.#.#
#...##........##........###.#
########################.....#
######..................######
#############################
```

Output for Sample Input 2

```
14 33
#################################
#################################
#####.........................######
#####.#################.###.######
#####.#...#######..###.#.#.######
#####.#...#####....###.###.######
#####.#.######..#..###.#.#.######
#####.########.....###.###.######
#####........#########.#.#.######
#####..................###.######
#####..................#.#.######
#####..................###.######
###########.................######
#################################
```

The second Sample Input encodes word "dabba", therefore the answer encodes word "abbad".