Problem 2 – Dragon Trap

Our statistics show that SoftUni students **LOVE** matrices and they also enjoy rotating stuff, that's why we decided to make you happy by giving you a problem in which you have to rotate elements inside a matrix. You're welcome!

So, what's the story? The town of Pernik is being harassed by a dragon. Meatball Sasho (a.k.a. "Sasho Kyufteto"), has the ultimate weapon, an elbow (a.k.a. "vinkel"), but he needs time to go home and take it. Meanwhile, the crafty townsfolk of Pernik need to distract the dragon by running in circles around him; that way, the dragon won't know which house or person to attack. Eventually Sasho will come to the rescue.

You'll be given a **rectangular matrix of characters** which represents the townsfolk – each person is represented by the first character of their name; Pernik is a weird place, so names can start with **any ASCII character**. On the first line of input you'll receive the **number of rows N** of the matrix and on each of the next N rows – a string representing the specified row.

On the next lines, until you receive the string **"End"**, you'll be given a series of commands in format: **"({row} {column}) {radius} {rotations}"**. **{row}** and **{column}** are the coordinates of the dragon (center of rotation), **{radius}** is the radius and **{rotations}** tells you how many times and in which direction to rotate the people (characters).

The top-left corner of the matrix has coordinates (0, 0). Rotation occurs only **within the matrix**, **on the outside of a square defined by the center and the radius** (check out the examples to grasp the concept). If the number of rotations is positive, rotate the characters **clockwise**, otherwise rotate them **counterclockwise**.

When the **"End"** command is given, print back the rotated matrix and on the last line, print how many cells have **changed values (compared to the original matrix)** after all rotations are complete in format: **"Symbols changed: {countOfSymbols}"**. Note that if two people are represented by the same character and one replaces the other, the cell doesn't change value.

Input

- The input data should be read from the console.
- On the first line, you'll be given a number **N** representing the number of rows of the matrix.
- On the next N lines you'll be given the rows of the matrix as strings.
- On the following lines, until the command **"End"** is received, you'll be given commands in the format described above.
- The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

- The output should be printed on the console. It should consist of **N + 1 lines**.
- On the first N lines, print back the **rotated matrix**.
- On the last line, print the number of cells which have changed values in the following format: **"Symbols changed: {countOfSymbols}"**.

Constraints

- The number of rows **N** of the matrix will be in the range [1 … 20].
- The matrix will contain ASCII characters.
- The number of rotating **commands** will be between 1 and 50, including the **"End"** command.
- The **{row}** and **{column}** will be integers in the range [-10 … 10].
- The **{radius}** will be an integer in the range [1 … 20].
- **{rotations}** will be an integer in the range $[-2^{31} \ldots 2^{31} - 1]$.
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input	Output	Comments
3 abc def	dab gec hif	The center is (1 1) which is the character 'e'. The radius is 1, so all shaded characters should be rotated. Note, we only rotate the 8 characters

ghi (1 1) 1 1 End	Symbols changed: 8	around 'e', but not the center (or any other cells inside the formed square). {rotations} is 1, which means we rotate the characters once clockwise: 'a' replaces 'b', 'b' replaces 'c', 'c' replaces 'f', etc.
3 abc bef ghi (-1 -1) 2 1 End	abc ebf ghi Symbols changed: 2	The square defined by the center and radius overlaps the matrix at the three shaded cells. Rotate once clockwise: one 'b' replaces the other 'b', and 'b' replaces 'e' and 'e' replaces a 'b'. Cell (0, 1) is still 'b', so in total, 2 symbols were changed.
3 abc def ghi (1 1) 2 1 End	abc def ghi Symbols changed: 0	The rotation square doesn't overlap the matrix, so no characters are rotated.