1. [M. Dynin, from a contest in St. Petersburg] Given a rectangle of letters (such as

```
AB
BC
```

), a starting position within that rectangle (such as the character in the upper-left corner), and a string (such as `"ABBC"`), consider the question of finding paths through the letters that match the given string, begin at the starting position, and at each step move one square in one of the eight compass directions (north, south, east, west, northeast, northwest, southeast, southwest). For the given example, there are two such paths. They are (`E-SW-E`) and (`S-NE-S`)—that is, "one step east, one step southwest, one step east" and "one step south, one step northeast, and one step south." For the rectangle

```
CBB
BBA
```

and the string `"BBCBBA"`, starting at square in the middle of the top row, there are 12 paths. Paths are allowed to visit the same position twice.

You are to write a program that, given such a rectangle, string, and starting position, reports the *number* of distinct paths that match the string, according to the definitions above. The input will consist of four positive integers (call them M, N, r, and c), a string (S), and M strings consisting of upper-case letters `A-Z`, each of which is N characters long. These inputs are all separated from each other by whitespace. The M strings are the rows of the rectangle, from top to bottom. The pair (r, c) are the coordinates of the starting position, with $0 \le r < M$, $0 \le c < N$. Row 0 is the top row; column 0 is the left column. The output will be a line reporting the number of paths in the format shown in the examples.

You may assume that the number of paths is less than 2^{31}, $M \le 80$, $N \le 80$. Nevertheless, be aware that the time limit is less than a minute.

Example 1:

Input	Output
2 2 0 0 ABBC AB BC	There are 2 paths.

Example 2:

Input	Output
2 3 0 1 BBCBBA CBB BBA	There are 12 paths.