Arch. ShadowArea

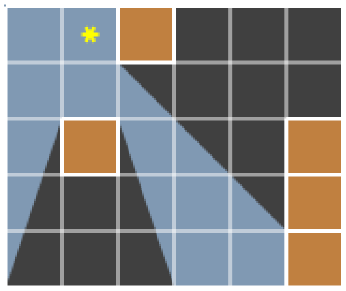
You will be given the layout of a rectangular room, where each character corresponds to a 1 meter by 1 meter square area of the room. A '`.`' (period) indicates an empty square, a '`#`' indicates a post that covers the square and extends from the floor to the ceiling, and a '`*`' (asterik) indicates a light source at the center of the square. There will be exactly 1 light source in the room.

The source will illuminate any point on the floor in an empty square if a straight line from to that point from the light source does not intersect any post. All other points in empty squares are in shadow. Given room, compute the area of the floor (in square meters) that is in shadow.

For example, consider the following room layout:

```
.*#...
......
.#...#
.....#
.....#
```

The post below the light casts a shadow of 5.0 square meters, and the shadow cast by the post to the right of the light has an area of 8.5 square meters, as shown in the figure below. Therefore, 13.5 is the correct answer.



Input

There are multiple test cases in the input file terminated by EOF. For each test case, first line contains two integers r, c ($1 \le r, c \le 50$) indicating the size of the room. Each of the next r lines contains a string of length c with the format described above.

Output

For each test case, output the desired number in a line.

Sample Input

```
5 6
.*#...
......
.#...#
.....#
.....#
10 30
..............................
..............................
..........#...................
.........#*#..................
..........#...................
..............................
..............................
..............................
..............................
..............................
3 7
.#....*
##.....
#......
6 10
..........
..........
..........
###..#####
..........
*.........
6 11
...........
...........
......#....
........#..
..........#
..........*
1 1
*
```

Sample Output

```
13.5
295.0
1.1666666666
29.2777777778
25.4333333333
0.0
```

Brav. TheSum

There is nothing more beautiful than just an integer number.

You are given integers a, b and c. Write each of them down in decimal notation with no leading zeros. The following operations can be performed on each of the written numbers:

- Insert a single digit at any position of the number (possibly at the beginning or at the end).
- Delete a single digit from the number.
- Replace a single digit in the number with some other digit.

An operation can only be performed if it results in a positive number with at least one digit and no leading zeros.

Each operation has an associated cost — *insCost*, *delCost* and *repCost*, respectively. Return the minimal possible total cost of operations needed to satisfy the equality $a + b = c$.

Input

There are multiple test cases in the input file terminated by EOF. For each test case, there are six integers $a, b, c, insCost, delCost, repCost$.
$(1 \le a, b, c \le 10^9; 0 \le insCost, delCost, repCost \le 10^6)$

Output

For each test case, please output the desired number.

Sample Input

```
44 77 11 1 1 1
44 77 11 4 4 1
100 100 10000 1000000 1000000 0
23456765 19876 927547301
7744 9876 9977
```

Sample Output

```
1
2
1000000
48697
```

(Almost empty page.)

Cinc. YetAnotherStonesGame

Johnny is playing a simple game with a set of colored stones and a circular board with squares laid out in a track around the edge, some of which are marked. To start with, he places some stones of different colors on the board in different unmarked squares and marks the starting position of each stone. He then picks some number from `allowedMoves` and advances some stone that number of squares clockwise around the board. If the square that the stone lands in is unmarked, then he leaves the stone there and marks the square. If the square was the starting position of that particular stone, then he removes the stone from the board. Otherwise, if the square had previously been marked, the move is invalid and he returns the stone to its position before he moved it at the beginning of this turn. He then repeats this process. The board is considered complete once all the squares are marked and all the stones have been removed (having moved at least once, then returned to their starting positions). To make the game more interesting, some of the squares on the board are marked before he starts the game and these squares are represented by a string, in which a marked square is denoted `'X'` and an unmarked square `'.'`. The squares are given in clockwise order. Note that since the board is circular, the first character is adjacent to the last character. Find out the minimum number of stones that Johnny would need to place on the board in order to complete the game, or -1 if it is impossible to complete the board.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

First line contains an integer n ($1 \leq n \leq 6$). Next n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 6$) representing the `allowedMove` array. Next line contains a string indicating the board with initial state.

Output

For each test case, please output the desired number.

Sample Input

```
1
4
............
1
4
.............
2
2 3
..X..XX.X...XX
3
5 3 6
....X...X.X.X..........X..X.....XXX....X....XXX
```

Sample Output

```
4
1
-1
3
```

(Almost empty page.)

Dweb. CircularShifts

You have two lists of numbers, `X`= $\{x_0, x_1, \ldots, x_{n-1}\}$ and `Y`= $\{y_0, y_1, \ldots, y_{n-1}\}$, each containing exactly n elements. You can optionally apply any number of circular shifts to each list. A circular shift means removing the last element from a list and re-inserting it before the first element. For example, {1, 2, 3} would become {3, 1, 2}, and {3, 1, 2} would become {2, 3, 1}. After you apply any circular shifts, the final score is calculated as:

$$x_0y_0 + x_1y_1 + \cdots + x_{n-1}y_{n-1}$$

You are given integers $Z0, A, B$ and M. Generate a list Z of length $2n$, using the following recursive definition:

- $Z(0) = Z0 \bmod M$
- $Z(i) = (Z(i-1) \times A + B) \bmod M$ (note that $Z(i-1) \times A + B$ may overflow a 32-bit integer)

Then, generate lists `X` and `Y`, each of length n, using the following definitions:

- $x_i = Z(i) \bmod 100$
- $y_i = Z(i+n) \bmod 100$

Return the maximal final score you can achieve.

Input

There are multiple test cases in the input file terminated by EOF. For each test case, there are 5 integers $n, Z0, A, B, M$. ($1 \le n \le 60000; 0 \le Z0, A, B \le 10^9; 1 \le M \le 10^9$)

Output

For each test case, please output the desired number.

Sample Input

```
5 1 1 0 13
4 1 1 1 20
10 23 11 51 4322
1000 3252 3458736 233421 111111111
141 96478 24834 74860 92112
```

Sample Output

```
5
70
28886
2585408
419992
```

(Almost empty page.)

Emol. DigitsSwap

You are given two positive integers, x and y, whose decimal representations contain the same number of digits. A digit-swap operation for an index i swaps the digits at the i-th positions in x and y. After exactly swaps digit-swap operations, what is the maximal possible value of $x \times y$? Return the String representation of this maximal product with no leading zeroes.

Input

There are multiple test cases in the input file terminated by EOF. For each test case, there are 3 integers x, y, S $(1 \leq x, y \leq 10^{50}; 0 \leq S \leq 10^9)$ indicating numbers x, y and number of swaps S. x and y will contain same number of digits.

Output

For each test case, please output the desired number.

Sample Input

```
123
321
2
4531
1332
0
13425
87694
99
2872876342876443222
2309482482304823423
5
940948
124551
4893846
56710852
18058360
1
9
1
1000000000
```

Sample Output

```
39483
6035292
1476187680
6669566046086333877050194232995188906
133434353148
1050671725722720
9
```

Frax. Unicorn

The unicorn is an exotic chess piece similar to the knight. The difference is that while the knight moves two cells in one direction and one in the other direction, the unicorn moves *more* than two cells in one direction and more than one in the other direction.

More precisely, each unicorn move looks as follows:

- pick up the unicorn;
- pick one of the four basic directions, and move the unicorn more than two cells in that direction;
- pick one of the two basic directions that are orthogonal to the previous one, and move the unicorn more than one cell in that direction;
- put down the unicorn.

We have a chessboard that has R rows times C columns. Each cell of the chessboard contains one of the first L letters of the alphabet. To keep the input small, the content of the cells is randomly generated using the method described below.

You are given the integers R, C, and L described above, an integer *seed* used to generate the letters in the cells, and a string *word*. You want to trace *word* by taking a unicorn, placing it onto a cell containing the first letter of *word*, making a valid move that takes it onto a cell that contains the second letter, from there making another move to the third letter, and so on. Find out the number of ways in which this can be done, modulo $1,000,000,007$.

The content of the cells is generated using the following pseudocode:

```
x = seed;
d = (65535 div L)+1;
for (int r=0; r<R; r++)
  for (int c=0; c<C; c++) {
    x = (x * 25173 + 13849) modulo 65536;
    chessboard[r][c] = character with ASCII code (65+(x div d));
  }
```

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

First line contains four integers $R, C, L, seed$. ($1 \leq R, C \leq 300$; $1 \leq L \leq 26$; $0 \leq seed \leq 65535$) Next line contains a string *word* with length between 1 and 50 inclusively, and composed of uppercase letters only.

Output

For each test case, please output the desired number.

Sample Input

```
3 4 2 47
AB
5 5 2 47
CD
4 4 1 42
AA
4 4 1 42
AAAAA
1 1 5 54321
ABCDE
8 8 6 226
TOPCODER
20 20 2 47
AAAAA
```

Sample Output

```
2
0
20
172
0
1
373977054
```

Giga. CompanyRestructuring

A small company has recently been growing quickly and has decided that it needs to restructure its organisation. In the past, there were no permanent managers and project teams were formed on an ad-hoc basis depending on which employees were available at the time. This has lead to a situation where many of the employees have managed each other at some time in the past. Now that the company is bigger, it wants to impose a more formal hierarchical structure. The company is to be divided into divisions, each of which will be structured as a tree. One employee in each division will be assigned as the division leader and can be the permanent manager of some other employees. These employees can in turn manage futher employees, and so on. Each employee will belong to exactly one division and each employee in a division other than the division leader will have a permanent manager in the division.

The company knows that such a restructuring may lead to ill-feeling amongst the employees, particularly if an employee ends up being managed permanently by another employee that he or she has managed in the past. It has therefore come up with the following model to try to minimise this ill-feeling. If an employee has managed a person in the past, then he or she feels superior to that person. This is also transitive, so if A has managed B, and B has managed C, then A feels superior to both B and C, even if A has not directly managed C. More formally, an employee X_0 feels superior to another employee X_k, if and only if there exists a sequence of employees $X_0, X_1, X_2, \ldots, X_k$, where employee X_i has managed employee X_{i+1} for each value of $i < k$. This can clearly lead to cases where each one of a pair of employees feels superior to the other and such a pair is termed *mutually superior*. These pairs of employees tend to spend a lot of time arguing about who is superior, so the company wishes to ensure they are separated in the new company structure. It has put the following restrictions on the structure of the new divisions.

- The new permanent manager of an employee must have managed that employee in the past.
- No employee can feel superior to his or her direct permanent manager.
- No pair of mutually superior employees can have the same direct permanent manager.

The company wishes to end up with as few divisions as possible. You are given a matrix, which details which employees have managed others in the past. The character in row i column j will be `'Y'` if employee i has managed employee j in the past and `'N'` otherwise. Return the minimum number of divisions that the company must create in order to satisfy the above restrictions.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

First line contains an integer n ($1 \leq n \leq 50$) indicating the number of employees. Each of the next n lines contain n characters as a string consisting of `'Y'` or `'N'`. The diagonal characters in the matrix is always `'N'`.

Output

For each test case, please output the desired number.

Sample Input

```
4
NNNN
NNYN
NNNN
YYYN
4
NNYN
NNYN
YNNN
YYYN
5
NYNNN
NNYNN
NNNYN
NNNNY
YNNNN
8
NYNYYYNN
NNNNYYNN
NYNNYYNN
NNYNYYNN
NNNNNNNN
NYYNYNNN
YYNYYYNN
YYNYYYYN
20
NYNYNNYYNYNNNYYNYNYY
YNNNNNYYNNNYYNYNYNYY
NNNNNNYYNNNYYNYNNNNY
YNYNNNNYNNNYNNYNYNNY
NYNNNNNYYYYNYNYYNNYN
YYYYNNYNYNNNNNNYYNYNY
NNNNNNNNNNNYYNNNNNYY
NNNNNNYNNNNYYNYNNNYN
NYYYNNNYNNNNYYNYYNYY
NNYNNNYYNNNNYNNNNNYY
YYNNNYNNYNNNNYNNYNYY
NNNNNNNYNNNNYNYNNNNY
NNNNNNYNNNNYNNNNNNNN
NNYNNNNNNNNYYNYNNYYN
NNNNNNNNNNNYNNNNNNNY
YNYYNYYNNNYYNNNNYNYY
NYNNNNNYNYNYYYYNNNNY
NNYNNNNYNYNYYYNNNNYY
NNNNNNNYNNNYNNNNNNNY
NNNNNNYYNNNYYNYNNNYN
```

(More samples are on the website.)

Sample Output

```
1
1
5
1
4
```

Herd. IncreasingLists

A string is called an increasing list if it is a comma-separated list of increasing positive integers with no leading zeroes. For example, the strings "`2,3,9`", "`30`", and "`1,100,1000000000000000000000`" are increasing lists, while "`5,6,6`", "`1,2,3,`", "`0`" and "`1,02`" are not.

You're given a String mask consisting of digits, commas and question marks. Replace each question mark with a digit or a comma, so that the resulting string is an increasing list, and return the resulting string. If there are multiple possible resulting strings, return the lexicographically smallest one. If it is impossible to produce an increasing list, return the string "`impossible`" (quotes for clarity only).

Input

There are multiple test cases in the input file terminated by EOF. For each test case, there is a string of length between 1 and 50 inclusively, and the string contains only characters comma, digits, and question marks.

Output

For each test case, please output the desired string.

Sample Input

```
??
???
?????????,9
??????????,9
?,10,?????????????????,16,??
?2?5??7?,??
???????????????????????????????,???
```

Sample Output

```
10
1,2
1,2,3,4,5,9
impossible
impossible
12,50,70,71
1,10,11,100,101,102,103,104,105,106
```

(Almost empty page.)

Ivys. PickingUp

There are two captains who want to form two teams of equal size. There are N players to choose from, and both captains have given a rating to each of the players. The ratings are given in the arrays of long longs `score1` and `score2`, where `score1[`i`]` is the first captain's rating of the i-th player, and `score2[`i`]` is the second captain's rating of the i-th player. The total score of a team is defined as the sum of all its players' ratings according to the team's captain. Players must be assigned to the two teams such that the difference between the teams' total scores is as small as possible. Output n integers, where the i-th element is the team to which the i-th player is assigned. The first captain's team is team 1, and the second captain's team is team 2. If multiple solutions exist, return the one that comes first lexicographically.

Input

There are multiple test cases in the input file terminated by EOF. For each test case:

First line contains an integer n. Next n integers indicating the array `score1`. Next n integers indicating the array `score2`.

Output

For each test case, please output the desired n integers in a line seperated by a space.

Sample Input

```
4
2 3 4 7
2 4 5 8
4
1 5 6 8
7 5 3 1
4
300 300 300 300
600 10 10 10
4
50 50 50 1
30 30 30 150
```

Sample Output

```
1 2 2 1
1 2 1 2
2 1 1 2
1 2 2 1
```

(Almost empty page.)

Jigg. SlimeXSlimeRancher

You are playing a game titled Slime Rancher. You will be training slimes in this game.

You have three slimes-in-training. Associated with each slime are N different attributes, each represented by a positive integer. You will be given the attributes of the slimes by three lists of N positive integers in order from first to last attribute.

You will train these slimes, and after the training is complete, each of the slimes' attributes will either increase by some integral value or stay the same. The weight of the training is defined as the sum of the differences between the final and initial values of all the attributes for all three slimes.

These slimes are going to be combined to make an even stronger slime. Three slimes produce the strongest result when combined if there exists a way such that when their final attributes are sorted in ascending order of values (identical valued attributes may be permuted to your liking), the attribute types match. That is, the i-th attribute in the sorted attributes for the first, second, and third slime must be the same attribute type (their values need not be the same, only their relative ordering).

You are a master slime breeder and you're able to obtain any possible final values for the slimes' attributes. What is the minimum possible weight of the training that gives the strongest result?

Input

There are multiple test cases in the input file terminated by EOF. For each test case, first line contains n. Next 3 lines contains lists of slimes of size n. $(1 \leq n \leq 150)$, all integers in the input file will be between 1 and 10^9. Watch out overflow.

Output

For each test case, output the desired result.

Sample Input

```
3
1 6 2
1 3 5
5 4 3
3
3 2 1
6 5 4
9 8 7
3
1 23 4
12 3 4
1 2 34
6
1 1 1 1000000000 1000000000 1000000000
1000000000 1000000000 1000000000 1 1 1
1 1 1 2 2 2
```

Sample Output

```
5
0
36
2999999997
```