11048 - Automatic Correction of Misspellings

Time limit: 3.000 seconds

Some text editors offer a feature to correct words which seem to be written incorrectly. In this problem you are asked to implement a simple *Automatic Correction of Misspellings (ACM)*.

ACM takes care of the following misspellings of words:

1. One letter is missing (e.g., **letter** is written **leter**)

   or too much (e.g., **letter** is written **lettter**).

2. One letter is wrong (e.g., **letter** is written **ketter**)
3. The order of two adjacent letters is wrong (e.g., **letter** is written **lettre**)

ACM is based on a dictionary of known words. When a text contains a word which is not in the dictionary, ACM will try to replace it by a similar word of the dictionary. Two words are similar if we can transform one word into the other by doing exactly one of the misspellings listed above. An unknown word is left unchanged if there is no similar word in the dictionary.

Input Specification

The first line of the input will give the number n of words in the dictionary ($n \leq 10000$). The next n lines contain the dictionary words. The following line will give the number of query words q ($q \leq 1000$). The next q lines contain the query words. You may assume that each word in the input consists of 1 to 25 lower case letters ('a' to 'z').

Output Specification

For each query word, print one line with the query word followed by one of the following possibilities:

1. **is correct**, if the word occurs in the dictionary.
2. **is a misspelling of <x>**, where <x> is a word of the dictionary similar to the query word, and the query word is not in the dictionary. In the case that there are several possibilities, select the word from the dictionary which appeared earlier in the input.
3. **is unknown**, if cases 1 and 2 do not apply.

Sample Input

```
10
this
is
a
dictionary
that
we
will
use
for
us
6
su
as
the
dictonary
us
willl
```

Sample Output

```
su is a misspelling of us
as is a misspelling of is
the is unknown
dictonary is a misspelling of
dictionary
us is correct
willl is a misspelling of
will
```

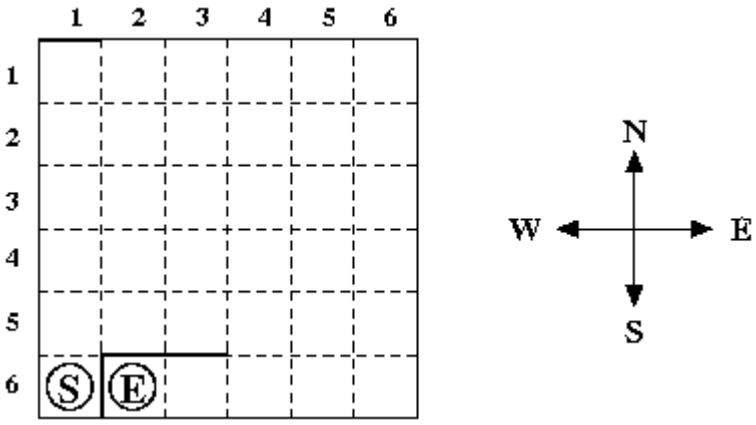
11049 - Basic wall maze

Time limit: 3.000 seconds

In this problem you have to solve a very simple maze consisting of:

1. a 6 by 6 grid of unit squares
2. 3 walls of length between 1 and 6 which are placed either horizontally or vertically to separate squares
3. one start and one end marker

A maze may look like this:



You have to find a shortest path between the square with the start marker and the square with the end marker. Only moves between adjacent grid squares are allowed; adjacent means that the grid squares share an edge and are not separated by a wall. It is not allowed to leave the grid.

Input Specification

The input consists of several test cases. Each test case consists of five lines: The first line contains the column and row number of the square with the start marker, the second line the column and row number of the square with the end marker. The third, fourth and fifth lines specify the locations of the three walls. The location of a wall is specified by either the position of its left end point followed by the position of its right end point (in case of a horizontal wall) or the position of its upper end point followed by the position of its lower end point (in case of a vertical wall). The position of a wall end point is given as the distance from the left side of the grid followed by the distance from the upper side of the grid.

You may assume that the three walls don't intersect with each other, although they may touch at some grid corner, and that the wall endpoints are on the grid. Moreover, there will always be a valid path from the start marker to the end marker. Note that the sample input specifies the maze from the picture above.

The last test case is followed by a line containing two zeros.

Output Specification

For each test case print a description of a shortest path from the start marker to the end marker. The description should specify the direction of every move ('N' for up, 'E' for right, 'S' for down and 'W' for left).

There can be more than one shortest path, in this case you can print any of them.

Sample Input

```
1 6
2 6
0 0 1 0
1 5 1 6
1 5 3 5
0 0
```

Sample Output

```
NEEESWW
```

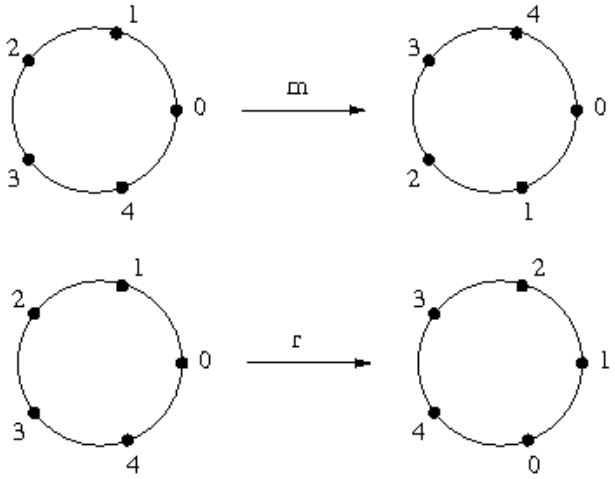
11051 - Dihedral groups

Time limit: 3.000 seconds

Consider n points on a unit circle with numbers $k = 0, 1, ..., n-1$. Initially point k makes an angle of $360 \cdot k / n$ degrees to the x-axis, measured in counter-clockwise direction. We are going to perform two different kind of operations on that set of points:

- rotation by $360 / n$ degrees in clockwise direction
- reflection with respect to the x-axis

The following picture shows an example of these operations:



Given a sequence of operations, we are interested in the shortest sequence of operations which gives the same result, i.e., the position of every single point is the same after performing either of those sequences of operations.

The sequence is given as a string consisting of the characters 'r' and 'm' which represent clockwise rotation and reflection respectively ("to the right" and "mirror"). Multiple consecutive occurrences of the same character are collected into the representation <character><number>, and for convenience this will also be done for single occurrences. So "rrmrrrrrrrrrrr" will be abbreviated to "r2 m1 r12". The representations of different operations are always separated by a single space.

Input Specification

The input consists of several test cases. Each test case starts with a line containing n ($3 \le n \le 10^8$), the number of points. The second line of each test case consists of an abbreviated sequence of operations as described above. All numbers will be positive and less than 10^8. There will be no empty line in the input, and no line will contain more than *100000* characters. The last test case is followed by a line containing 0.

Output Specification

For each test case print one line containing the abbreviated format of the sequence with the minimum number of operations which results in the same configuration of points as the input sequence. In case of multiple optimal solutions, print any solution.

Sample Input

```
4
r2
100
m1 r100 m1
54
r218 m3 r1
0
```

Sample Output

```
r2

r1 m1
```

Note that the second line of the sample output is a blank line

11054 - Wine trading in Gergovia

Time limit: 3.000 seconds

As you may know from the comic "Asterix and the Chieftain's Shield", Gergovia consists of one street, and every inhabitant of the city is a wine salesman. You wonder how this economy works? Simple enough: everyone buys wine from other inhabitants of the city. Every day each inhabitant decides how much wine he wants to buy or sell. Interestingly, demand and supply is always the same, so that each inhabitant gets what he wants.

There is one problem, however: Transporting wine from one house to another results in work. Since all wines are equally good, the inhabitants of Gergovia don't care which persons they are doing trade with, they are only interested in selling or buying a specific amount of wine. They are clever enough to figure out a way of trading so that the overall amount of work needed for transports is minimized.

In this problem you are asked to reconstruct the trading during one day in Gergovia. For simplicity we will assume that the houses are built along a straight line with equal distance between adjacent houses. Transporting one bottle of wine from one house to an adjacent house results in one unit of work.

Input Specification

The input consists of several test cases. Each test case starts with the number of inhabitants n ($2 \le n \le 100000$). The following line contains n integers a_i ($-1000 \le a_i \le 1000$). If $a_i \ge 0$, it means that the inhabitant living in the i^{th} house wants to buy a_i bottles of wine, otherwise if $a_i < 0$, he wants to sell $-a_i$ bottles of wine. You may assume that the numbers a_i sum up to 0. The last test case is followed by a line containing 0.

Output Specification

For each test case print the minimum amount of work units needed so that every inhabitant has his demand fulfilled. You may assume that this number fits into a signed 64-bit integer (in C/C++ you can use the data type "long long", in JAVA the data type "long").

Sample Input

```
5
5 -4 1 -3 1
6
-1000 -1000 -1000 1000 1000 1000
0
```

Sample Output

```
9
9000
```

11055 - Homogeneous squares

Time limit: 3.000 seconds

Assume you have a square of size n that is divided into $n \times n$ positions just as a checkerboard. Two positions (x_1,y_1) and (x_2,y_2), where $1 \leq x_1,y_1,x_2,y_2 \leq n$, are called "independent" if they occupy different rows and different columns, that is, $x_1 \neq x_2$ and $y_1 \neq y_2$. More generally, n positions are called independent if they are pairwise independent. It follows that there are $n!$ different ways to choose n independent positions.

Assume further that a number is written in each position of such an $n \times n$ square. This square is called "homogeneous" if the sum of the numbers written in n independent positions is the same, no matter how the positions are chosen. Write a program to determine if a given square is homogeneous!

Input Specification

The input contains several test cases.
The first line of each test case contains an integer n ($1 \leq n \leq 1000$). Each of the next n lines contains n numbers, separated by exactly one space character. Each number is an integer from the interval $[-1000000,1000000]$.
The last test case is followed by a zero.

Output Specification

For each test case output whether the specified square is homogeneous or not. Adhere to the format shown in the sample output.

Sample Input

```
2
1 2
3 4
3
1 3 4
8 6 -2
-3 4 0
0
```

Sample Output

```
homogeneous
not homogeneous
```

10307 - Killing Aliens in Borg Maze

Time limit: 3.000 seconds

The Borg is an immensely powerful race of enhanced humanoids from the delta quadrant of the galaxy. The Borg collective is the term used to describe the group consciousness of the Borg civilization. Each Borg individual is linked to the collective by a sophisticated subspace network that insures each member is given constant supervision and guidance.

Your task is to help the Borg (yes, really) by developing a program which helps the Borg to estimate the minimal cost of scanning a maze for the assimilation of aliens hiding in the maze, by moving in north, west, east, and south steps. The tricky thing is that the beginning of the search is conducted by a large group of over **100** individuals. Whenever an alien is assimilated, or at the beginning of the search, the group may split in two or more groups (but their consciousness is still collective.). The cost of searching a maze is defined as the total distance covered by all the groups involved in the search together. That is, if the original group walks five steps, then splits into two groups each walking three steps, the total distance is **11=5+3+3**.

Input

On the first line of input there is one integer, **N <= 50**, giving the number of test cases in the input. Each test case starts with a line containing two integers **x, y** such that **1 <= x, *y* <= 50**. After this, **y** lines follow, each which **x** characters. For each character, a space `` '' stands for an open space, a hash mark ``**#**'' stands for an obstructing wall, the capital letter ``**A**'' stand for an alien, and the capital letter ``**S**'' stands for the start of the search. The perimeter of the maze is always closed, i.e., there is no way to get out from the coordinate of the ``**S**''. At most **100** aliens are present in the maze, and everyone is reachable.

Output

For every test case, output one line containing the minimal cost of a successful search of the maze leaving no aliens alive.

Sample Input

```
2
6 5
#####
#A#A##
# # A#
#S  ##
#####
7 7
#####
#AAA###
#     A#
# S ###
#       #
#AAA###
#####
```

Sample Output

```
8

11
```