3646 - Transcript

Europe - Southwestern - 2006/2007

Transcript

A company wants to hire a new employee. The selection process consists of several attention tests, one of them consists in: each candidate must type each character he sees in a flashing screen, using a given keyboard. You are asked to write down a program to score the candidates, given the original sequence of characters, and what the candidate actually typed. Scoring is based on the kind of actions the candidate may perform. For each character flashing in the screen, she may only:

1. Correctly type the character;

2. Omit the character;

3. By mistake, type a different character.

The score or penalty given to each action depends on the keyboard layout considered. The keyboard is a matrix of *n* rows and *m* columns.

The **distance** between the characters at coordinates (i_1,j_1) and (i_2,j_2) is given by the maximum of $|i_1- i_2|$ and $|j_1-j_2|$. For example, in the keyboard below (3 rows by 5 columns), the distance between the character "a", at (2,1), and the character "h" at (3,5) is 4, and the distance between the character "o" and "h" is 2. In this example, the largest distance between any two characters is 4. For any keyboard the largest distance between any two characters is conventionally referred to by TOP.

$$\begin{array}{cccccc} 1 & \mathbf{e} & \mathbf{g} & \mathbf{y} & \mathbf{i} & \mathbf{m} \\ 2 & \mathbf{a} & \mathbf{n} & \mathbf{o} & \mathbf{w} & \mathbf{s} \\ 3 & \mathbf{u} & \mathbf{f} & \mathbf{l} & \mathbf{t} & \mathbf{h} \\ & 1 & 2 & 3 & 4 & 5 \end{array}$$

The score given to the correct transcription of one character is TOP+1. The penalty given for the omission of one character (action 2) is TOP+1. The score given for changing a character for another (action 3) is TOP+1 minus the distance between the two characters involved in the mistake. For example, considering the keyboard shown above, the following action scores apply: Score for the correct transcript of one character: 5; Penalty for the omission of one character: 5; Score for changing character "o" to character "h": 5-2=3.

A scoring of a test is the sum of the scores given to each character typed minus the sum of the penalties for each character omitted. Since the scoring is only done after the test finishes, it is not possible to be sure about when specific actions were realized (e.g, did the candidate skip a character, or mistyped it?). To avoid complaints, the final score given is the **highest possible value for scoring** the candidate, according to the rules explained above. For example, if the text to transcribe is "time" and the candidate types "yme", we may score it in several ways:

To omit "t", change "i" for "y" and correctly transcribe "m" and "e";

To change "t" for "y", omit "i" and correctly transcribe "m" and "e";

To change "t" for "y", change "i" for "m", omit "m" and correctly transcribe "e";

To change "t" for "y", change "i" for "m", change "m" for "e" and omit "e".

Each one of these possibilities has one score associated (9, 8, 7, and 3 points, respectively). Thus, the candidateÕs final score is 9 points. Write a program that, considering the shape of a keyboard, the text to be transcribed by the candidate, and the actual transcript produced by the candidate returns the final score of the candidate.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N, stating the number of rows in the keyboard. The next N lines, all with the same length, contain a string with the sequence of characters in the corresponding keyboard row. The keyboard will not have more than 20 rows and 30 columns. Then, the next two lines contains the text to be transcribed by the candidate, and the text typed by the candidate. These texts will be no longer than 500 characters each.

Output

For each test case, a single line containing an integer stating the final score of the candidate.

Sample Input

3

egyim

anows

uflth

time

yme

Sample Output

9

Southwestern 2006-2007

3656 - Odd or Even

Latin America - South America - 2006/2007

There are several versions of Odd or Even, a game played by competitors to decide random issues (such as ``who will code this problem?"). In one of the versions, for two players, the game starts with each player calling either odds or evens. Then they count to three (some people chant ``Once, twice, three, SHOOT!"). On three, both players hold out one of their hands, showing a number of fingers (from zero to five). If the fingers add to an even number, then the person who called evens wins. If the fingers add to an odd number, then the person who called odds wins.

John and Mary played several games of Odd or Even. In every game John chose odds (and, consequently, Mary chose evens). During the games each player wrote down, in small cards, how many fingers he/she showed, using one card for each game - Mary used blue cards, John used red cards. Their objective was to be able to re-check the results later, looking at the cards for each game. However, at the end of the day John dropped the deck of cards, and although they could separate the cards by color, they are now out of order.

Given the set of numbers written on red cards and on blue cards, you must write a program to determine the minimum number of games that Mary certainly won.

Input

The input contains several test cases. The first line of a test case contains an integer N representing the number of games played ($1 \le N \le 100$). The second line of a test case contains N integers X_i , indicating the number of fingers shown by Mary in each of the games ($0 \le X_i \le 5$, for $1 \le i \le N$). The third line of a test case contains N integers Y_i , indicating the number of fingers shown by John in each of the games ($0 \le Y_i \le 5$, for $1 \le i \le N$). The end of input is indicated by $N = 0$.

Output

For each test case your program must write one line, containing one integer, indicating the minimum number of games that Mary certainly won.

Sample Input

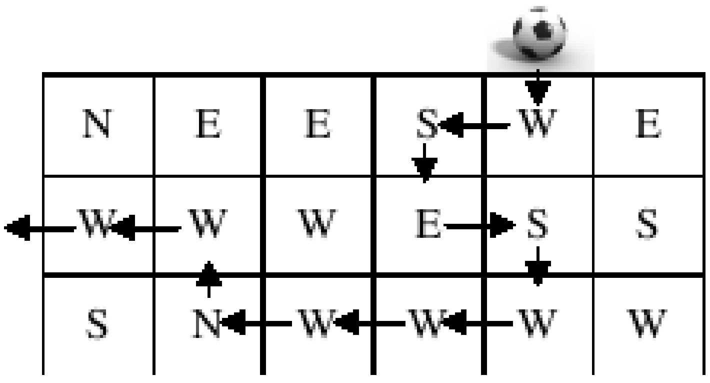
```
3
1 0 4
3 1 2
9
0 2 2 4 2 1 2 0 4
1 2 3 4 5 0 1 2 3
0
```

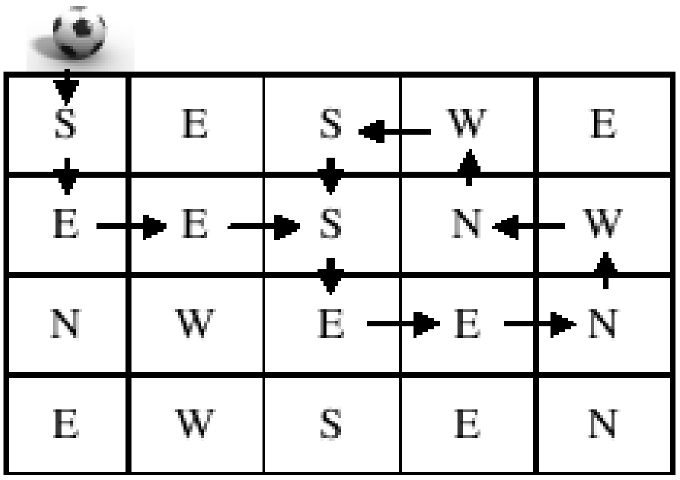
Sample Output

```
0
3
```

3687 - Football Foundation (FOFO)

Latin America - Mexico and Central America - 2006/2007


GRID 1


GRID 2

The football foundation (FOFO) has been researching on soccer; they created a set of sensors to describe the ball behavior based on a grid uniformly distributed on the field. They found that they could predict the ball movements based on historical analysis. Each square sensor of the grid can detect the following patterns:

`N` north (up the field)

`S` south (south the field)

`E` east (to the right on the field)

`W` west (to the left on the field)

For example, in grid 1, suppose the ball was thrown into the field from north side into the field. The path the sensors detected for this movement follows as shown. The ball went through 10 sensors before leaving the field.

Comparing with what happened on grid 2, the ball went through 3 sensors only once, and started a loop through 8 instructions and never exits the field.

You are selected to write a program in order to evaluate line judges job, with the following out put based on each grid of sensors, the program needs to determine how long it takes to the ball to get out of the grid or how the ball loops around.

Input

There will be one or more grids of sensors for the same game. The data for each is in the following form. On the first line are three integers separated by blanks: The number of rows in the grid, the number of columns in the grid, and the number of the column in which the ball enters from the north. The grid column's number starts with one at the left. Then come the rows of direction instructions. The lines of instructions contain only the characters `N`, `S`, `E` or `W`, with no blanks.

The end of input is indicated by a grid containing `` `0 0 0' `` as limits.

Output

For each grid in the input there is one line of output. Either the ball follows a certain number of sensors and exits the field on any one of the four sides or else the ball follows the behavior on some number of sensors repeatedly. The sample input below corresponds to the two grids above and illustrates the two forms of output. The word ``step" is always immediately followed by ``(s)" whether or not the number before is 1.

Sample Input

```
3 6 5
NEESWE
WWWESS
SNWWWW
4 5 1
SESWE
EESNW
NWEEN
EWSEN
0 0 0
```

Sample Output

```
10 step(s) to exit
3 step(s) before a loop of 8 step(s)
```

Mexico and Central America 2006-2007

3708 - Graveyard

Europe - Northeastern Europe & Russian Republic - 2006/2007

Programming contests became so popular in the year 2397 that the governor of New Earck -- the largest human-inhabited planet of the galaxy -- opened a special Alley of Contestant Memories (ACM) at the local graveyard. The ACM encircles a green park, and holds the holographic statues of famous contestants placed equidistantly along the park perimeter. The alley has to be renewed from time to time when a new group of memorials arrives.

When new memorials are added, the exact place for each can be selected arbitrarily along the ACM, but the equidistant disposition must be maintained by moving some of the old statues along the alley.

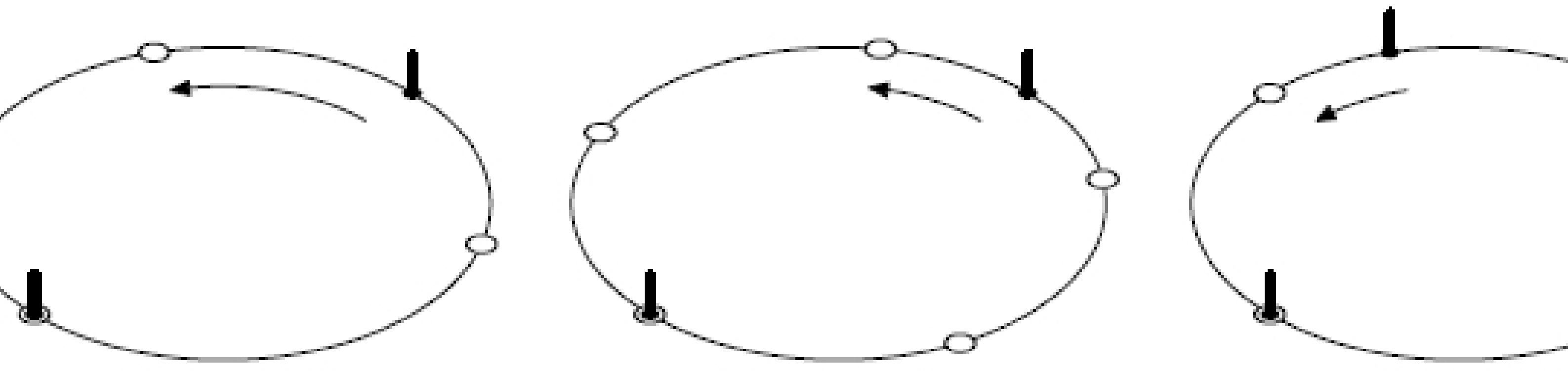
Surprisingly, humans are still quite superstitious in 24th century: the graveyard keepers believe the holograms are holding dead people souls, and thus always try to renew the ACM with minimal possible movements of existing statues (besides, the holographic equipment is very heavy). Statues are moved along the park perimeter. Your work is to find a renewal plan which minimizes the sum of travel distances of all statues. Installation of a new hologram adds no distance penalty, so choose the places for newcomers wisely!

Input

The input file contains several test cases, each of them consists of a a line that contains two integer numbers: n -- the number of holographic statues initially located at the ACM, and m -- the number of statues to be added ($2 \le n \le 1000$, $1 \le m \le 1000$) . The length of the alley along the park perimeter is exactly 10 000 feet.

Output

For each test case, write to the output a line with a single real number -- the minimal sum of travel distances of all statues (in feet). The answer must be precise to at least 4 digits after decimal point.



Pictures show the first three examples. Marked circles denote original statues, empty circles denote new equidistant places, arrows denote movement plans for existing statues.

Sample Input

```
2 1
2 3
3 1
10 10
```

Sample Output

```
1666.6667
1000.0
1666.6667
0.0
```

Northeastern Europe & Russian Republic 2006-2007

3986 - The Bridges of Kolsberg

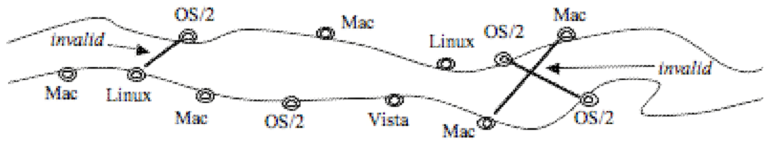
Europe - Southwestern - 2007/2008

Problem F - The Bridges of K lsberg

Background

King Beer has a very hard region to rule, consisting of lots of cities with very sectarian operating system beliefs and high levels of trade. These cities are placed along a river, the K lsberg, along its Northern and Southern banks. The cities are economically separated from each other, since the river is wide and dangerous.

invalid
invalid



A section of the K lsberg showing some *<u>invalid</u>* *bridges*

King Beer would like to build some bridges connecting opposite banks of the river. He was strongly advised against making bridges between cities with different operating systems beliefs (those guys really hate each other). So, he is just going to build bridges between cities sharing the same operating system belief (even if the resulting bridges are quite long and strangely shaped). However, it is technical impossible to build bridges that cross other bridges.

The economical value of a bridge is the sum of the trade values the two cities it connects. The King wants to maximize the sum of all possible bridge values while minimizing the number of bridges to build.

Problem

Given two sets of cities, return the maximum possible sum of all bridge values and the smallest number of valid bridges necessary to achieve it.

Input

The first line is an integer with the number of samples. For each sample, the next line has a non-negative integer, not greater than 1,000, indicating the number of cities on the Northern riverbank. Then, on each line, comes the city information with the form

cityname ostype tradevalue

where, separated by empty spaces, there are two strings, *cityname* and *ostype*, with no more than 10 characters each, and *tradevalue* which is a non-negative integer not greater than 10^6. The sequence of lines represents the cities from left to right along the riverbank. Next, there is the same kind of information to describe the Southern riverbank.

Output

For each sample, a line consisting of the maximum possible sum of all bridge values, one empty space, the number of bridges.

Sample Input

```
1
3

mordor Vista 1000000

xanadu Mac 1000

shangrila OS2 400

4

atlantis Mac 5000

hell Vista 1200

rivendell OS2 100

appleTree Mac 50
```

Sample Output

```
1002250 2
```

Southwestern 2007-2008

Problem G: Line & Circle Maze

Source file: `maze.{c, cpp, java}`
Input file: `maze.in`

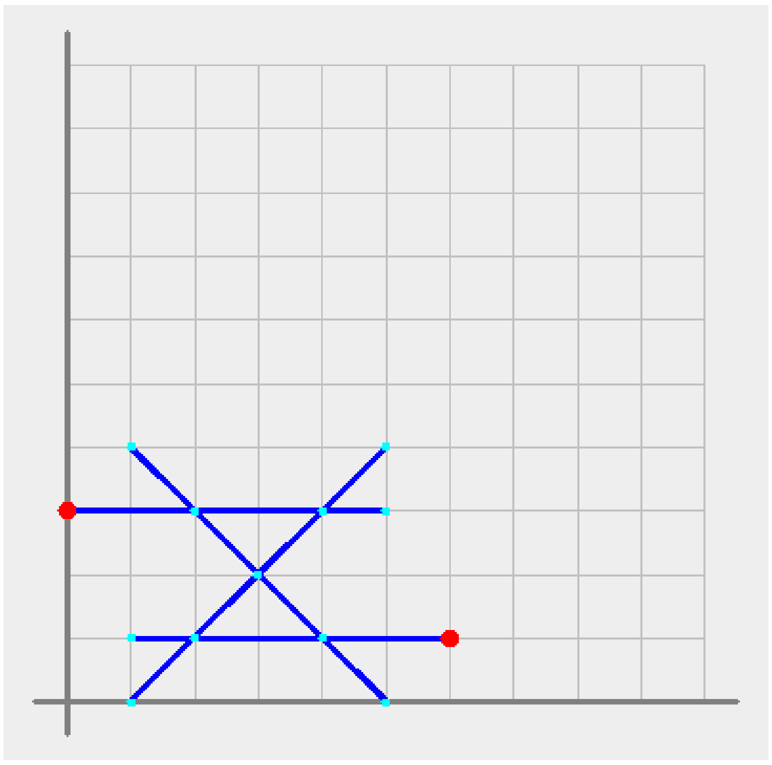
A deranged algorithms professor has devised a terrible final exam: he throws his students into a strange maze formed entirely of linear and circular paths, with line segment endpoints and object intersections forming the junctions of the maze. The professor gives his students a map of the maze and a fixed amount of time to find the exit before he floods the maze with xerobiton particles, causing anyone still in the maze to be immediately inverted at the quantum level. Students who escape pass the course; those who don't are trapped forever in a parallel universe where the grass is blue and the sky is green.

The entrance and the exit are always at a junction as defined above. Knowing that clever ACM programming students will always follow the shortest possible path between two junctions, he chooses the entrance and exit junctions so that the distance that they have to travel is as far as possible. That is, he examines all pairs of junctions that have a path between them, and selects a pair of junctions whose shortest path distance is the longest possible for the maze (which he rebuilds every semester, of course, as the motivation to cheat on this exam is very high).

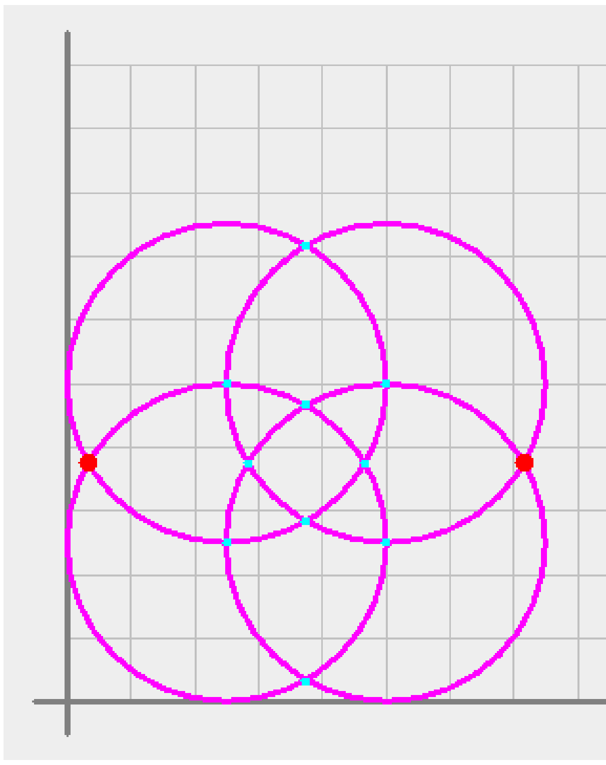
The joy he derives from quantumly inverting the majority of his students is marred by the tedium of computing the length of the longest of the shortest paths (he needs this to know to decide how much time to put on the clock), so he wants you to write a program to do it for him. He already has a program that generates the mazes, essentially just a random collection of line segments and circles. Your job is to take that collection of line segments and circles, determine the shortest paths between all the distinct pairs of junctions, and report the length of the longest one.

The input to your program is the output of the program that generates his mazes. That program was written by another student, much like yourself, and it meets a few of the professor's specifications: 1) No endpoint of a line segment will lie on a circle; 2) No line segment will intersect a circle at a tangent; 3) If two circles intersect, they intersect at exactly two distinct points; 4) Every maze contains at least two junctions; that is, a minimum maze is either a single line segment, or two circles that intersect. There is, however, one bug in the program. (He would like to have it fixed, but unfortunately the student who wrote the code never gave him the source, and is now forever trapped in a parallel universe.) That bug is that the maze is not always entirely connected. There might be line segments or circles, or both, off by themselves that intersect nothing, or even little "submazes" composed of intersecting line segments and circles that as a whole are not connected to the rest of the maze. The professor insists that your solution account for this! The length that you report must be for a path between connected junctions!

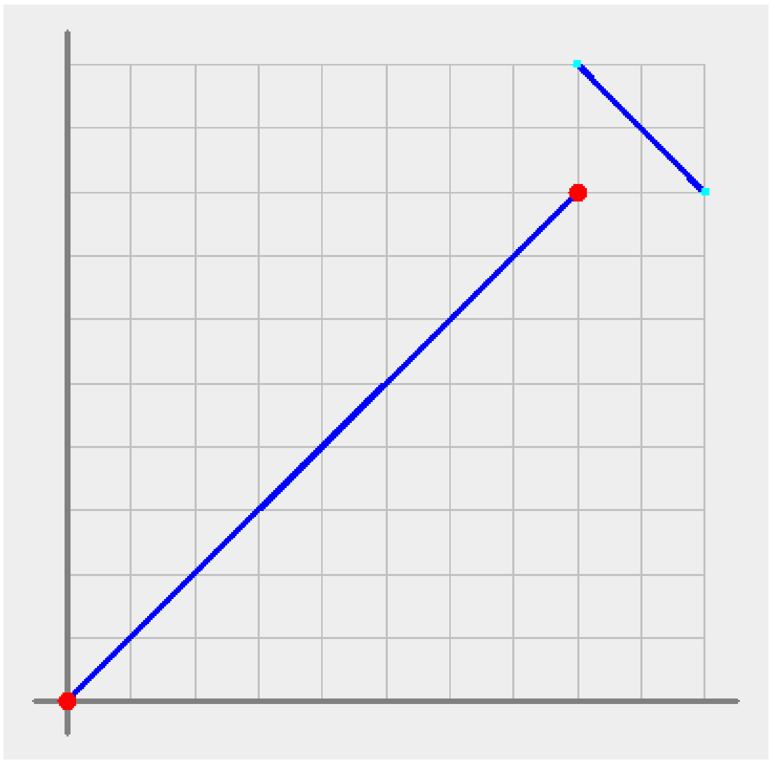
Examples:



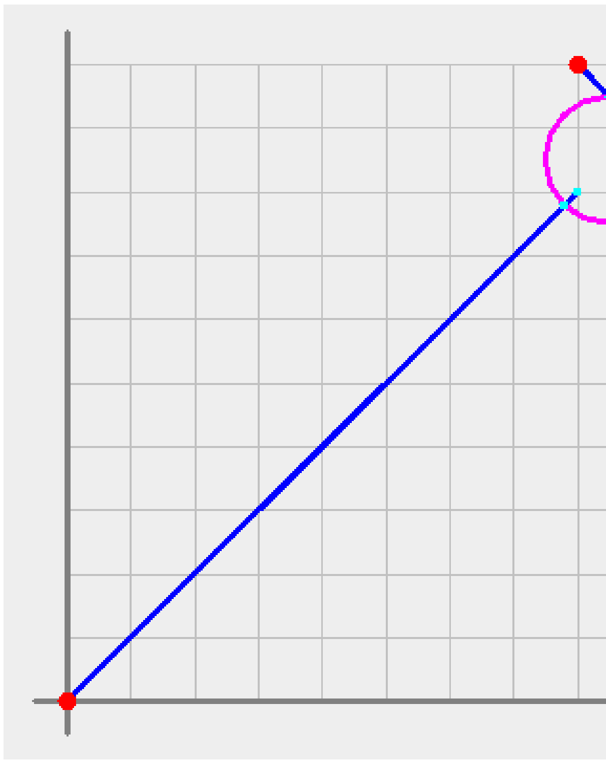
Line segments only. The large dots are the junction pair whose shortest path is the longest possible.



An example using circles only. Note that in this cas also another pair of junctions with the same length possible shortest path.



Disconnected components.



Now the line segments are connected by a circle, al longer shortest path.

Input: An input test case is a collection of line segments and circles. A line segment is specified as "L X_1 Y_1 X_2 Y_2" where "L" is a literal character, and (X_1,Y_1) and (X_2,Y_2) are the line segment endpoints. A circle is specified by "C X Y R" where "C" is a literal character, (X,Y) is the center of the circle, and R is its radius. All input values are integers, and line segment and circle objects are entirely contained in the first quadrant within the box defined by (0,0) at the lower left and (100,100) at the upper right. Each test case will consist of from 1 to 20 objects, terminated by a line containing only a single asterisk. Following the final test case, a line containing only a single asterisk marks the end of the input.

Output: For each input maze, output "Case N: ", where N is the input case number starting at one (1), followed by the length, rounded to one decimal, of the longest possible shortest path between a pair of connected junctions.

Example Input:	**Example Output:**

```
L 10 0 50 40
L 10 40 50 0
L 10 10 60 10
L 0 30 50 30
*
C 25 25 25
C 50 25 25
C 25 50 25
C 50 50 25
*
L 0 0 80 80
L 80 100 100 80
*
L 0 0 80 80
L 80 100 100 80
C 85 85 10
*
*
```
	```
	Case 1: 68.3
	Case 2: 78.5
	Case 3: 113.1
	Case 4: 140.8
	```

Note: It is recommended that the atan2() function, rather than acos() or asin(), be used when calculating arc angles.

Last modified on October 26, 2008 at 2:15 PM.

Mid Central 2008-2009

4211 - Bases

Latin America - South America - 2008/2009

What do you get if you multiply 6 by 9? The answer, of course, is 42, but only if you do the calculations in base 13.

Given an integer $B \geq 2$, the *base B numbering system* is a manner of writing integers using only digits between 0 and B -1, inclusive. In a number written in base B, the rightmost digit has its value multiplied by 1, the second rightmost digit has its value multiplied by B, the third rightmost digit has its value multiplied by B^2, and so on.

Some equations are true or false depending on the base they are considered in. The equation 2+2=4, for instance, is true for any $B \geq 5$ - it does not hold in base 4, for instance, since there is no digit '4' in base 4. On the other hand, an equation like 2+2=5 is never true.

Write a program that given an equation determines for which bases it holds.

Input

Each line of the input contains a test case; each test case is an equation of the form "`EXPR=EXPR`", where both "`EXPR`" are arithmetic expressions with at most 17 characters.

All expressions are valid, and contain only the characters '`+`', '`*`' and the digits from '`0`' to '`9`'. No expressions contain leading plus signs, and no numbers in it have leading zeros.

The end of input is indicated by a line containing only "`=`".

Output

For each test case in the input your program should produce a single line in the output, indicating for which bases the given equation holds.

If the expression is true for infinitely many bases, print "`B+`", where B is the first base for which the equation holds.

If the expression is valid only for a finite set of bases, print them in ascending order, separated by single spaces.

If the expression is not true in any base, print the character '`*`'.

Sample input	**Output for the sample input**

`6*9=42`
`10000+3*5*334=3*5000+10+0`
`2+2=3`
`2+2=4`
`0*0=0`
`=`
	`13`
	`6 10`
	`*`
	`5+`
	`2+`

4295 - Videopoker

Europe - Northwestern - 2008/2009

Videopoker is the slot machine variant of the currently immensely popular game of poker. It is a variant on draw poker. In this game the player gets a hand consisting of five cards randomly drawn from a standard 52-card deck. From this hand, the player may discard any number of cards (between 0 and 5, inclusive), and change them for new cards randomly drawn from the remainder of the deck. After that, the hand is evaluated and the player is rewarded according to a payout structure. A common payout structure is as follows:

Hand	Payout
one pair	1
two pair	2
three of a kind	3
straight	4
flush	5
full house	10
four of a kind	25
straight flush	100
royal flush	250

Once you know the payout structure, you can determine for a given hand which cards you must change to maximize your expected reward. We'd like to know this expected reward, given a hand.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with nine integers x_i ($0 \leq x_i \leq 1000$\$) describing the payout structure. The numbers are in increasing order and describe the payout for one pair, two pair, etc, until the royal flush.
- One line with one integer n ($1 \leq n \leq 10$): the number of starting hands to follow.
- n lines, each describing a starting hand. A hand consists of five space separated tokens of the form `Xs`, with `X` being the rank (`` `2' `` \dots `` `9' ``, `` `T' ``, `` `J' ``, `` `Q' ``, `` `K' `` or `` `A' ``) and `s` being the suit (`` `c' ``, `` `d' ``, `` `h' `` or `` `s' ``).

Output

Per testcase:

- One line for each starting hand with a floating point number that is the maximal expected reward for that hand. These numbers must have an absolute or relative error less than 10^{-6}.

Sample Input

```
1
1 2 3 4 5 10 25 100 250
5
Ah Ac Ad As 2s
Ks Qs Js Ts 2h
Ks Qs 2d 2h 3s
2d 4h 5d 3c 9c
2h 3h 6d 8h Tc
```

Sample Output

```
25.000000
8.9574468
1.5467160
0.9361702
0.6608135
```

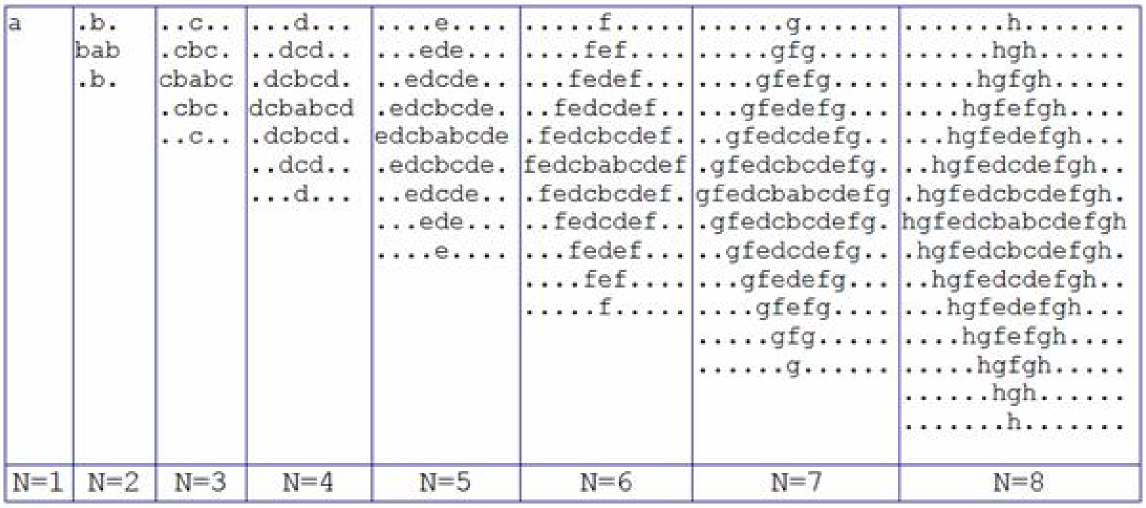
The 2008 ACM Northwestern European Programming Contest

Northwestern 2008-2009

4403 - ASCII Diamondi

Asia - Kuala Lumpur - 2008/2009

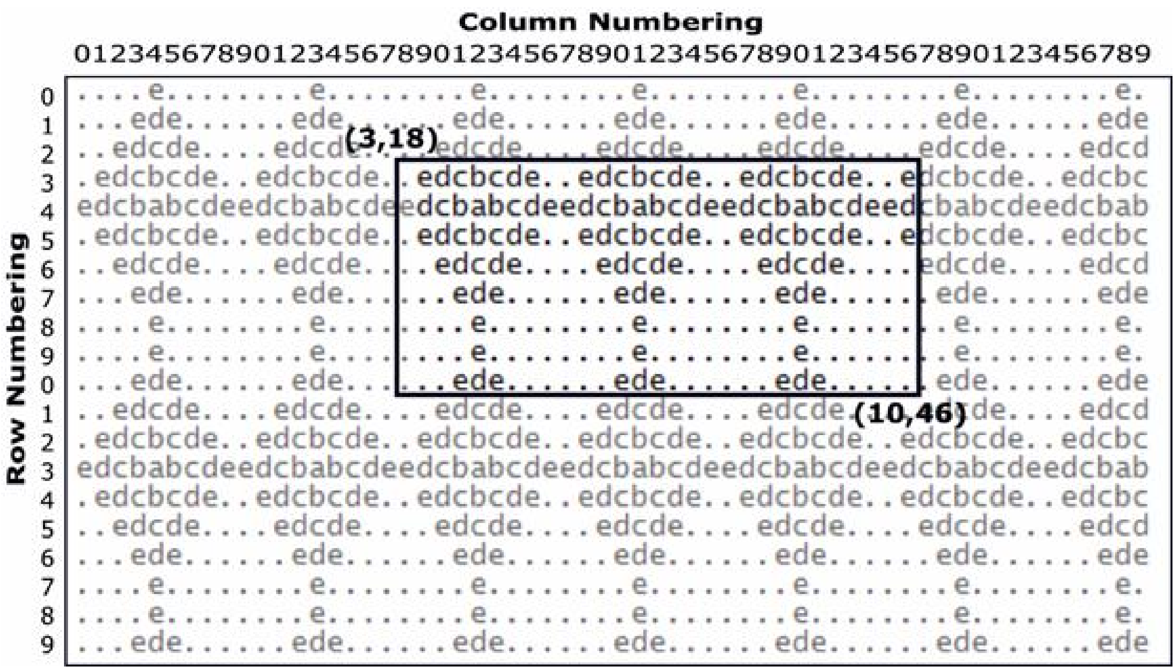
ASCII diamonds can be drawn with integer side lengths. Each layer of this diamond is drawn with a single ASCII alphabet, starting with **a** and ending with **z** (starting from the center) and continues in cyclic order.



a	.b.	..c..	...d...	....e....	.....f.....	......g......	.......h.......
	bab	.cbc.	..dcd..	...ede...	....fef....	.....gfg.....	......hgh......
	.b.	cbabc	.dcbcd.	..edcde..	...fedef...	....gfefg....	.....hgfgh.....
		.cbc.	dcbabcd	.edcbcde.	..fedcdef..	...gfedefg...	....hgfefgh....
		..c..	.dcbcd.	edcbabcde	.fedcbcdef.	..gfedcdefg..	...hgfedefgh...
			..dcd..	.edcbcde.	fedcbabcdef	.gfedcbcdefg.	..hgfedcdefgh..
			...d...	..edcde..	.fedcbcdef.	gfedcbabcdefg	.hgfedcbcdefgh.
				...ede...	..fedcdef..	.gfedcbcdefg.	hgfedcbabcdefgh
				....e....	...fedef...	..gfedcdefg..	.hgfedcbcdefgh.
					....fef....	...gfedefg...	..hgfedcdefgh..
					.....f.....	....gfefg....	...hgfedefgh...
						.....gfg.....	....hgfefgh....
						......g......	.....hgfgh.....
							......hgh......
							.......h.......
N=1	N=2	N=3	N=4	N=5	N=6	N=7	N=8

Figure 1: ASCII diamond for different side lengths.

Any one of these **ASCII** diamonds can be used to draw an infinite plane by using this as a tile. For example **ASCII** diamond of length **5** can be used to draw such an infinite grid. Only first **20** row and **60** columns are shown below:



```
  012345678901234567890123456789012345678901234567890123456789
0 ....e........e........e........e........e........e........e.
1 ...ede......ede......ede......ede......ede......ede......ede
2 ..edcde....edcde....edcde....edcde....edcde....edcde....edcd
3 .edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbc
4 edcbabcdeedcbabcdeedcbabcdeedcbabcdeedcbabcdeedcbabcdeedcbab
5 .edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbc
6 ..edcde....edcde....edcde....edcde....edcde....edcde....edcd
7 ...ede......ede......ede......ede......ede......ede......ede
8 ....e........e........e........e........e........e........e.
9 ....e........e........e........e........e........e........e.
0 ...ede......ede......ede......ede......ede......ede......ede
1 ..edcde....edcde....edcde....edcde....edcde....edcde....edcd
2 .edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbc
3 edcbabcdeedcbabcdeedcbabcdeedcbabcdeedcbabcdeedcbabcdeedcbab
4 .edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbcde..edcbc
5 ..edcde....edcde....edcde....edcde....edcde....edcde....edcd
6 ...ede......ede......ede......ede......ede......ede......ede
7 ....e........e........e........e........e........e........e.
8 ....e........e........e........e........e........e........e.
9 ...ede......ede......ede......ede......ede......ede......ede
```

Here rows and columns are numbered starting from zero. By specifying the topmost row (**row$_1$**), leftmost column (**col$_1$**), bottommost row (**row$_2$**) and rightmost column (**col$_2$**) we can specify a portion of such an infinite grid (also shown in figure above).

Given the side length of the tile to be used, the topmost row (**row$_1$**), leftmost column (**col$_1$**), bottommost row (**row$_2$**) and rightmost column (**col$_2$**) you have to print the pattern within these four boundaries (inclusive).

Input

Input contains at most **125** sets of inputs. But not all cases are extreme.

Each set of input contains five integers: **N (0<N£20000)**, **row$_1$**, **col$_1$**, **row$_2$**, **col$_2$** **(0£ row$_1$ £ row$_2$ £ 20000, 0 £ col$_1$ £ col$_2$ £ 20000, 0£ (row$_2$- row$_1$+1)* (col$_2$- col$_1$+1) £ 40000)**. Here **N** denotes that the side length of the tiles used to draw the plane should be **N**. The meaning of **row$_1$, col$_1$, row$_2$, col$_2$** are given in the problem statement. The first sample input corresponds to the figure above.

Input is terminated by a line where the first integer is zero.

Output

For each line of input produce **(row$_2$- row$_1$+2)** lines of output. First line contains serial of output. Each of the next lines contain **(col$_2$- col$_1$+1)** characters. These lines describe the patterns within the specified rows and columns. Look at the output for sample input for details. The output file size is less than **1** MB.

Sample Input

```
5 3 18 10 46

100 50 50 69 69

0 2 3 4 5
```

Output for Sample Input

```
Case 1:

.edcbcde..edcbcde..edcbcde..e

edcbabcdeedcbabcdeedcbabcdeed

.edcbcde..edcbcde..edcbcde..e

..edcde....edcde....edcde....
```

```
...ede......ede......ede.....
....e........e........e......
....e........e........e......
...ede......ede......ede.....
Case 2:
utsrqponmlkjihgfedcb
tsrqponmlkjihgfedcba
srqponmlkjihgfedcbaz
rqponmlkjihgfedcbazy
qponmlkjihgfedcbazyx
ponmlkjihgfedcbazyxw
onmlkjihgfedcbazyxwv
nmlkjihgfedcbazyxwvu
mlkjihgfedcbazyxwvut
lkjihgfedcbazyxwvuts
kjihgfedcbazyxwvutsr
jihgfedcbazyxwvutsrq
ihgfedcbazyxwvutsrqp
hgfedcbazyxwvutsrqpo
gfedcbazyxwvutsrqpon
fedcbazyxwvutsrqponm
edcbazyxwvutsrqponml
dcbazyxwvutsrqponmlk
cbazyxwvutsrqponmlkj
bazyxwvutsrqponmlkji
```

Problem setter: Shahriar Manzoor, Special Thanks: Sohel Hafiz, Md. Arifuzzaman Arif

Kuala Lumpur 2008-2009

Problemsetter: Shahriar Manzoor
Special Thanks: Md. Arifuzzaman Arif, Sohel Hafiz

4405 - Tariff Plan

Asia - Kuala Lumpur - 2008/2009

Ampang Communications & Mobile (ACM) provides telecom services for various types of users. Since the people of Ampang are quite talkative, they are always seeking for packages that are best suited for them. To have an edge over their competitors, ACM provides various packages. Two of the most popular packages are:

- Mile
- Juice

Mile charges every **30** seconds at a rate of **10** cents. That means if you talk for **29** seconds or less, you will be charged with **10** cents. If you talk for **30** to **59** seconds, you will be charged with **20** cents and so on.

Juice charges every **60** seconds at a rate of **15** cents. That means if you talk for **59** seconds or less, you will be charged with **15** cents. Similarly, if you talk for **60** seconds to **119** seconds, you will be charged with **30** cents and so on.

Given a list of call durations, can you determine the package that is cheaper?

Input

The first line of input is an integer **T(T<50)** that denotes the total number of test cases. Each case starts with a line containing an integer **N(0<N<20)**. The next line gives a list of **N** call durations (In second). Each call duration is an integer in the range **[1, 2000]**. Consecutive integers are separated by a single space character.

Output

For each case, output the case number first. Then output the name of the cheaper package followed by the corresponding cost in cents. If both package gives the same total cost, then output both the names (**Mile** preceding **Juice**) followed by the cost. Look at the output for sample input for details.

Sample Input

3

2

61 10

3

40 40 40

2

60 65

Output for Sample Input

Case 1: Mile 40

Case 2: Juice 45

Case 3: Mile Juice 60

Problem setter: Sohel Hafiz, Special Thanks: Shamim Hafiz, Rujia Liu

Illustration: Case 1: **Mile**(30+10=40) & **Juice**(30+15=45). Case 2: **Mile**(20+20+20=60) & **Juice**(15+15+15=45). Case 3: **Mile**(30+30=60) & **Juice**(30+30=60).

Kuala Lumpur 2008-2009

Problemsetter: Sohel Hafiz
Special Thanks: Rujia Liu, Shamin Hafiz