SWERC 2006

Contest Session

19 November 2006

9 Problems

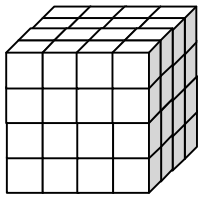
DEPARTAMENTO DE INFORMÁTICA

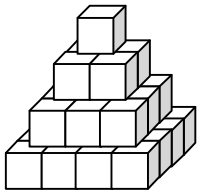
Problem A Cubes$^{\text{squared}}$

Cube Factory Ltd is an enterprise that sells *hcubes* (short of "harmonic cubes"), a very fashionable item nowadays providing great profits for his owner, Mr. Tesseract (his friends call him Mr.T).

Mr. T just bought a very large space to fit his increasing stock of *hcubes*. *Hcubes* have a plain cube format and are not hard to stock. However, Mr. T has a (rather harmless?) mania: he only admits two valid ways to pile them: (a) in cube format or (b) in squared pyramids (i.e., where each new step holds an increasing square number of elements).

One example of each type (holding, respectively, 4^3=64 *hcubes* and $1^2+2^2+3^2+4^2$ = 30 *hcubes*):





Given N *hcubes*, find the minimal number of valid piles to stock them according to Mr. T rules.

Example: to stock 38 *hcubes* we only need two piles: e.g., one cube of height 2 (holding 8 *hcubes*) and a pyramid of height 4 (holding 30 *hcubes*).

Input

The input file contains several lines. Each line consists of a single integer representing the number N of *hcubes* ($0 \le N \le 400.000$). The file ends in a line with the number –1.

Output

For each N in the input file, a line containing the corresponding result.

Sample Input

```
38
60
12
39101
-1
```

Sample Output

```
2
2
4
4
```

Problem B BlindFold

A labyrinth is a set of interconnected rooms. Each room has a few doors, each door is labelled, just on one side, by either “A”, “B”, “C” or “D”. There might be more than one door in a room with the same label, and each door in a room always leads to a room (sometimes the same room!) in the labyrinth, through a dark and convoluted tunnel. Moreover, doors may only be opened from their labelled side, so after getting through a door to another room, you may only use the new labelled doors you may find there.

Given two labyrinths L1 and L2, and a room R1 of L1 and R2 of L2, we say that R1 is equivalent to R2 if the two following conditions hold:

1. For each door labelled with *x* in R1 one may choose a door labelled with the same label *x* in R2 such that the two doors lead to equivalent rooms R1’ and R2’ (of L1 and L2 respectively).
2. For each door labelled with *x* in R2 one may choose a door labelled with the same label *x* in R1 such that the two doors lead to equivalent rooms R2’ and R1’ (of L2 and L1 respectively).

Notice that if R1 and R2 are equivalent rooms then the set of door labels in R1 is the same as the set of door labels of R2.

We say that the labyrinths L1 and L2 are equivalent if their initial rooms are equivalent. We are asked to write a program that checks whether two labyrinths are equivalent according to the definition above.

Input

The input file specifies the structure of the two labyrinths, one after the other. The specification of each labyrinth is as follows. First, an integer N indicates the total number of doors in the labyrinth. Then, for each each door, a line of the form ***i a d*** follows, where ***i*** is an integer indicating the initial room, ***a*** is the door label (one of “A”, “B”, “C”, “D”) and ***d*** is an integer indicating the destination room. Different doors in a room may well be labelled with the same letter. By convention, the visitor starts the visit in room 1.
The labyrinths considered will not have more than 200 rooms. Rooms are numbered 1,2, …, etc.

Output

A single line containing “`yes`” if the two labyrinths are equivalent, and “`no`” if they are not equivalent.

Sample Input A

```
2
1 A 2
2 A 1
3
1 A 1
1 A 2
2 A 2
```

Sample Output A

```
yes
```

Sample Input B

```
4
1 A 2
2 B 1
2 A 3
3 C 1
5
1 A 2
1 A 4
2 A 3
3 C 1
4 B 1
```

Sample Output B

```
no
```

Problem C X-Plosives

A secret service developed a new kind of explosive that attain its volatile property only when a specific association of products occurs. Each product is a mix of two different simple compounds, to which we call a *binding pair*. If N>2, then mixing N different binding pairs containing N simple compounds creates a powerful explosive**.** For example, the binding pairs A+B, B+C, A+C (three pairs, three compounds) result in an explosive, while A+B, B+C, A+D (three pairs, four compounds) does not.

You are not a secret agent but only a guy in a delivery agency with one dangerous problem: receive binding pairs in sequential order and place them in a cargo ship. However, you must avoid placing in the same room an explosive association. So, after placing a set of pairs, if you receive one pair that might produce an explosion with some of the pairs already in stock, you must refuse it, otherwise, you must accept it.

An example. Let's assume you receive the following sequence: A+B, G+B, D+F, A+E, E+G, F+H. You would accept the first four pairs but then refuse E+G since it would be possible to make the following explosive with the previous pairs: A+B, G+B, A+E, E+G (4 pairs with 4 simple compounds). Finally, you would accept the last pair, F+H.

Compute the number of refusals given a sequence of binding pairs.

Input

Instead of letters we will use integers to represent compounds. The input file contains several lines. Each line (except the last) consists of two integers (each integer lies between 0 and 10^5) separated by a single space, representing a binding pair. The file ends in a line with the number –1. You may assume that no repeated binding pairs appears in the input.

Output

A single line with the number of refusals.

Sample Input

```
1 2
3 4
3 5
3 1
2 3
4 1
2 6
6 5
-1
```

Sample Output

```
3
```

Problem D Berlin

The administration of a well-known football team has made a study about the lack of support in international away games. This study has concluded that the only reason for this lack of support is the difficulty in organizing the travel arrangements. To help solving this problem, the administration has asked you to build a program that computes the maximum number of people that can fly from a departure city to a destination city, using the available places in regular flights in a given day, and arriving at or before a given time. When traveling from one city to another, a person may make multiple transfers. Each transfer is, at least, 30 minutes long, i.e., the departure time should be, at least 30 minutes after the arrival time. Given a set of flights for a single day and the latest arrival time, your program should compute the maximum number of persons that can fly, directly or indirectly, from a departure city to a destination city, arriving at or before the latest arrival time.

Input

The first line contains an integer (smaller or equal to 150) indicating the number of cities that have flight connections. The second line contains a string indicating the city of departure. The third line contains a string indicating the destination city. The fourth line contains the latest arrival time, in the format HHMM, where HH is the hour in the day (from 00 to 23) and MM is the minute in the hour (from 00 to 59). The fifth line contains an integer N (smaller or equal to 5000), with the number of existing flights. Each of the following N lines contains the info for each flight. Each such line contains two strings and three integers, separated by blank spaces, O E C D A, where O and E are, respectively, the origin and destination of a flight, C is the number of available places in the flight (from 0 to 300), and D and A are the departure and arrival times in the previously defined format HHMM. All flights start and end in the same day. City names may have up to 8 characters.

Output

The output consists of one single line with an integer stating the maximum number of people that can fly from the origin to the destination city, using the given flights and arriving at or before the given latest arrival time.

Sample Input

```
4
lisbon
berlin
1500
9
lisbon london 6 1000 1100
london lisbon 6 1130 1230
lisbon paris 5 1000 1100
paris lisbon 4 1130 1230
london paris 1 1130 1300
london berlin 2 1340 1510
berlin london 2 1300 1430
paris berlin 10 1330 1500
berlin paris 9 1300 1430
```

Sample Output

```
6
```

Problem E 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.

1	e	g	y	i	m
2	a	n	o	w	s
3	u	f	l	t	h
	1	2	3	4	5

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 candidate's 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 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

A single line containing an integer stating the final score of the candidate.

Sample Input

```
3
egyim
anows
uflth
time
yme
```

Sample Output

```
9
```

Problem F Tip

During the recent Football Worldcup, a group of friends worked at a courtyard café to pay for their holidays. Everyday they would collect all the tips given by the customers on a jar, and at the end of the day they wanted to split the tips equally between them. After a few days, they reached the conclusion that (given the various face values of euro coins) sometimes it was not possible to equally split the collect of the day between them.

Write a program to help the friends determine if it is possible (or not) to equally split the collect of the day between them.

Input

The input will consist of a sequence of pairs of lines, each pair represents a coin division problem to be solved. For each such pair the first line contains the number of friends (a positive integer not greater than 5). The second line contains eight space separated non-negative integers n_1, n_2,…, n_8, where n_i is the number of coins of value i (0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00 and 2.00 euros respectively, e.g., the number of 5 cents coins will be denoted by n_3). The maximum number of coins is 10000. Input is terminated by a single line with the number -1.

Output

For each coin division problem print either "`yes`" or "`no`", depending on whether it is possible or not to divide equally the tips by the friends.

Sample Input

```
2
1 1 1 1 1 1 1 1
2
2 1 2 1 5 2 2 1
1
3423 234 324 972 740 12 234 901
4
147 5502 3486 434 76 66 267 20
5
3015 3590 1559 1219 78 507 23 8
-1
```

Sample Output

```
no
yes
yes
no
yes
```

Problem G Booby Traps

Tomb raiders are used to explore complex labyrinths searching treasures. Nowadays, they do it more effectively with the help of hi-tech. However, a team of Chinese tomb raiders have found labyrinths so complex that they can't handle them using their usual techniques. These labyrinths are made of cells, many of them with traps. If one moves into a cell with a trap, then the trap gets triggered. Moreover, if a trap gets triggered, its cell becomes blocked, and cannot be used anymore in a path. To make things worse, they have discovered that traps may have effects at a distance on other traps: if one triggers a trap, then traps associated to it also become triggered. They also discovered that the association between traps comes from a "domination ordering" defined on trap kinds, in such a way that triggering a trap of kind α causes all traps of kind "less than or equal to" α in the domination ordering to become also triggered. You are asked to write a program that given a labyrinth such as this, indicates the minimum number of moves needed to reach the end position from the start position. A move consists in going up, down, left or right (no diagonal moves).

Input

The input contains the domination ordering on the trap kinds, the map, and the start and end positions. The first line of the input contains a string with 26 distinct characters defining the domination order of the trap kinds. The first character indicates the weakest trap kind (that only triggers its own kind), the last character is the strongest (that triggers traps of all kinds). The second line contains two integers that indicate the width (w) and height (h) of the map, where $h \times w$ do not exceed 40000. The following h lines of the input contain w symbols that define the cells of the map (separated by an empty space). The following codes are used: '`x`' represents a cell with solid wall, a symbol α from '`A`' to '`Z`' identifies a cell containing a trap of kind α, and '`o`' represents an empty cell. The two last lines define the coordinates of the start and end positions, with two integers for the x (column) and y (line) coordinates in the grid, starting from 0.

Output

If the input describes a problem with a solution, then the output must be an integer in a single line indicating the minimum number of moves needed to reach the end position from the start position. If no solution exists, then the output should contain just "`IMPOSSIBLE`" in a single line.

Sample Input

```
ZYXWVUTSRQPONMLKJIHGFEDCBA
8 8
x x x x o x x x
x o o C o o o x
x A x x x x D x
x o o o o o o x
x x x o x x x x
x o o B o o o x
x A x x x B x x
o o o o o o o o
4 0
4 7
```

Sample Output

```
17
```

Problem H Fire Lane

The ambulances of Lilliput City Fire Department often have trouble to get across a particularly busy crossroads in the down town. Frequently, ambulance drivers have to shout orders asking drivers to remove cars from a certain lane in the street, a special lane reserved for emergency vehicles, the "Fire Lane".

To model the situation, we consider an imaginary grid representing the street. Vehicles occupy one or more cells in this grid. The vehicles always move forward (no turns), towards the exterior of the grid, and towards a definite direction (N,S,W,E). Unfortunately, there might be other vehicles blocking their way. Your objective is to compute the necessary orders to clear the Fire Lane of those blocking vehicles so that the ambulance can pass through.

The Fire Lane consists of two distinguished consecutive columns in the grid. Consider, for instance, the following grid:

```
. 0 0 0 0 . . .
. 4 1 1 1 . . .
. 4 . . . . . .
. . . 3 3 3 . .
. . 2 . . . . .
. . 2 . . . . .
. . 2 . . . . .
. . . . . . . .
```

In this picture, the symbols '`.`' denote free cells, the numbers mark the vehicles, and the gray shadow identify columns 3 and 4 as being the Fire Lane. Suppose that vehicles `0` and `3` are moving East (right), vehicle `1` is moving West (left), vehicle `2` is moving South (down), and vehicle `4` is moving North (up). Your program must compute which vehicles must be moved away, and list them. In this example, the vehicles to be moved are just 0, 3, 4 and 1, in this order.

Input

The first line of the input contains two integers indicating the size of the grid (width and height up to 100). Positions in the grid are represented by integer coordinates x and y, starting from 0, where x denotes the column and y the row, with origin the upper left corner. The second line of the input contains an integer N that indicates the number of vehicles to be listed. The third line contains an integer that indicates the x coordinate of the left column of the Fire Lane (which always has 2 columns). The next N lines give information for each of the N vehicles in the grid. For each line, describing a vehicle, there are two integers x, y indicating its coordinates in the grid, an integer for the length of the vehicle (the vehicles are 1 cell wide), and a character for the direction (`N`, `S`, `W` or `E`) towards which the vehicle extends and may move, starting from its position x, y. The numeric code of a vehicle is given by its position in the input sequence, starting with 0.

Output

The output of your program is a list of vehicle numbers, each number in a separate line, to present a solution, or the word "`Jammed`" if there are no solutions to the given problem. In case there is a solution, it should refer just to vehicles that need indeed to be removed. To list all the vehicles to be removed, you should at each step select

among the vehicles that may be moved away (i.e., that are not blocked) the one with the least code number.

Sample Input

```
8 8
5
3
1 0 4 E
4 1 3 W
2 4 3 S
3 3 3 E
1 2 2 N
```

Sample Output

```
0
3
4
1
```

Problem I Gap

There are many different ways of saying things and also of writing things. Sometimes there are much more different ways of saying things than things that are worth to be said. Sometimes people invent funny ways of telling things, sometimes people find complicated ways of not telling anything. This is true not only of people professionally engaged in the communication business, but also of everyone else. It is hard to communicate (do you see what we mean?).

To help you correlate what some people may want to say (or not) you decided to workout a tool that may be used to compare two texts with gaps, and tell whether the patches may be filled so that the resulting texts will become the same. Gaps in the text are named by strings and have a length attached. For example, the sequence of characters `<firstname:10>` identifies a gap named "firstname" of length 10. Here is an example of texts with gaps (we identify blank spaces in the input with the symbol ␣).

```
<name:3>'s␣boat␣is␣no␣longer␣than␣Anne's.␣If␣Joe␣likes
␣<food:7>␣then␣so␣do␣I.␣Usually,␣<food:7>␣are␣yellow.

<thing:10>␣is␣no␣longer␣than␣Anne's.␣If␣<name:3>␣likes
␣bananas␣then␣so␣do␣I.␣Usually,␣bananas␣are␣<color:6>.
```

The texts may be matched by consistently replacing each named gap by some string of the appropriate length. A gap with a given name may appear in either text, possibly several times, always with the same associated length. In the example above, we get:

```
color yellow
food bananas
name Joe
thing Joe's boat
```

Your goal is to write a program that given two texts with gaps will determine if the texts can be matched, in which case it must list how to fill the gaps, or not.

Input

The input consists of two texts, in sequence. A text is given as a sequence of printable characters, for convenience split into several lines. Each text is specified by an integer N, in a single line, indicating how many lines the text has, followed by precisely N lines. To obtain the text from its lines, you should just concatenate the contents of all the lines, in order. The text may contain the usual alphabetic and punctuation characters, and also sequences of the form `<`*`identifier`*`:`*`integer`*`>` indicating a gap in the text. The *`identifier`* is a sequence of alphabetic characters, with the name of the gap, and *`integer`* is an integer (between 0 and 32), with the length of the gap. The number of lines in each test does not exceed 100 lines, and each line does not exceed 400 characters.

Output

If the texts can be matched, the output will contain "`yes`" in the first line, followed by the strings that have been chosen to fill the gaps. For each gap, you should list the gap identifier, followed by a single space, and the text selected to fill the given gap. This list should appear by alphabetic order of the gap identifiers. If your matching is not able to identify some character precisely, then such character must be printed as "`*`" in the output.

If the texts cannot be matched, the output should contain "`no`" in a single line.

Sample Input A

```
2
<name:3>'s␣boat␣is␣no␣longer␣than␣Anne's.␣If␣Joe␣likes
␣<food:7>␣then␣so␣do␣I.␣Usually,␣<food:7>␣are␣yellow.
2
<thing:10>␣is␣no␣longer␣than␣Anne's.␣If␣<name:3>␣likes
␣bananas␣then␣so␣do␣I.␣Usually,␣bananas␣are␣<color:6>.
```

Sample Output A

```
yes
color yellow
food bananas
name Joe
thing Joe's boat
```

Sample Input B

```
1
potato<bingo:6>.
1
po<bobo:6> ppp.
```

Sample Output B

```
yes
bingo ** ppp
bobo tato**
```