PROBLEM 8 – CLOCK SYNCHRONISATION

Consider a set of *C* clocks. Each clock has a single hour hand that can only point to one of the following hours: 3, 6, 9, or 12. These clocks can be moved in *B* different ways called block moves. Each block move applies to a specified subset of clocks, and consists of moving the hand of each clock in the subset clockwise by 3 hours. Your task is to find a minimal sequence of block moves that sets all clocks to 12 hours. Multiple repetitions of a single block move are allowed. If two solutions of the same length exist, then give the lesser one in dictionary order. If there is no solution then write "`No solution.`" If no move is required then write "`No moves required.`"

For example, consider a set of 9 clocks (*C* = 9) with initial positions as shown in the following table, and 9 available block moves (*B* = 9), where ticks indicate the clocks affected by each block move:

Clock numbers	0	1	2	3	4	5	6	7	8
Clock positions	9	9	12	6	6	6	6	3	6
Block move 0	√	√		√	√				
Block move 1	√	√	√						
Block move 2		√	√		√	√			
Block move 3	√			√			√		
Block move 4		√		√	√	√		√	
Block move 5			√			√			√
Block move 6				√	√		√	√	
Block move 7							√	√	√
Block move 8					√	√		√	√

With clocks in the initial positions shown in the above table, applying block move 3 would result in the following updated clock positions (clocks 0, 3, and 6 have been moved clockwise by 3 hours):

Clock numbers	0	1	2	3	4	5	6	7	8
Clock positions	12	9	12	9	6	6	9	3	6

Further applying block moves 4, 7, and 8 would result in all clocks being set to 12 hours. The move sequence 3, 4, 7, 8 is actually the required solution in this case.

INPUT FORMAT

Input consists of a number of scenarios. Each scenario starts with a line containing a scenario title, which is a string of 1 to 20 letters, digits, and underscores (with no intervening spaces). A single # on a line indicates the end of input.

The "title" line is followed by one line consisting of two integers *C* and *B*, separated by a single space:

- *C* is the number of clocks, $1 \leq C \leq 11$, where the clocks are numbered sequentially starting with 0 (i.e., 0, 1, 2, 3, … *C*-1),
- B is the number of blocks, $0 \leq B \leq 11$, where the blocks are numbered sequentially starting with 0 (i.e., 0, 1, 2, 3, … *B*-1).

This line is followed by one line representing the starting positions of our clocks. This line consists of *C* clock positions separated by single spaces, each position being an integer number in the set {3, 9, 6, 12}.

This "clocks" line is followed by *B* other lines, one line for each block move operation. Each "block" line consists of 1 or more clock numbers (in no particular order), separated by single spaces, representing the clocks that will advance together by 3 hours in this block.

SAMPLE INPUT:

```
Test_1_0
1 0
12
Test_1_1
1 1
12
0
Test_3_3
3 3
9 9 9
0 1
1 2
0 2
Test_9_9
9 9
9 9 12 6 6 6 6 3 6
1 3 4 0
0 1 2
2 4 5 1
0 3 6
4 5 7 1 3
2 5 8
3 4 6 7
8 7 6
8 7 4 5
#
```

OUTPUT FORMAT

Each set of output data consists of a single output line showing in order: the problem title, a colon (“:”), a space (“ “), and one of the following answers:

- the text “`No moves required.`”, if no moves are required to solve the problem,
- the text “`No solution.`”, if the problem doesn’t have any solution,
- the block numbers making up the minimal solution (in the required length/dictionary order), separated by single spaces.

SAMPLE OUTPUT:

```
Test_1_0: No moves required.
Test_1_1: No moves required.
Test_3_3: No solution.
Test_9_9: 3 4 7 8
```

PROBLEM 7 – SPREADING GOSSIP

We have a remote village with $n \leq 20$ houses (h_0, h_1, h_2, … h_{n-1}) and several secure telephone lines linking neighbouring houses (there is exactly one line between each pair of neighbouring houses). For any pair of houses h_i and h_j there is at least one path of telephone lines connecting them (this can be viewed as an undirected connected graph with houses as vertices and lines as edges).

Gossip can travel over telephone lines. Each house can call at most one neighbour house at a time. Calls may begin at the beginning of each hour (e.g., 9 am, 1 pm, 6 pm, etc), and last for exactly one hour. The local telephone company charges a fortune for each call, but has a quirk that any number of calls can be made in parallel at the same price as any single call.

Given this scenario, we want to find the minimum total price (minimum number of used hours) to disseminate some gossip from house h_0 to all other houses.

INPUT FORMAT

The input involves a series of scenarios. Within each scenario the first line has an integer number n, the number of houses. This first line is followed by n other lines, one for each house, in the order h_0, h_1, h_2, … h_{n-1}. Each "house" line contains a list of indices of its neighbouring houses (in no particular order), separated by single spaces.

The series is terminated by a scenario with n=0, which isn't processed.

SAMPLE INPUT:

```
4
1 2
0 3
3 0
1 2
7
1 2 3
0 2
0 1 3 4
0 2
6 2 5
4 6
4 5
0
```

OUTPUT FORMAT

The output must be "`Village s: p`", where *s* is the scenario sequence number starting at 1 and *p* is the answer for each input village.

SAMPLE OUTPUT:

```
Village 1: 2
Village 2: 4
```

PROBLEM 1 – HOUSE NUMBERING

The government of Acmonia has decided that henceforth all house numbers should be given in binary instead of decimal notation. Householders will now have to purchase 0 and 1 binary digits to display on their houses. For reasons much too complicated to discuss here it seems that the cost to a householder of a 0 binary digit and of a 1 binary digit may well differ. Your task is to write a program which will report to householders the cost of their new numbers.

INPUT FORMAT

The input text consists of a number of sets of problems. The first line of a set is of the form "`COST a b`". For that set:

- a and b are both integers, $0 \le a, b \le 1000$,
- a 0 binary digit costs a dollars,
- a 1 binary digit costs b dollars.

The first line is followed by one or more lines each consisting of a single integer n.

- $0 \le n \le 2{,}000{,}000$,
- n indicates a house number, expressed as a standard decimal number.

A single `#` on a line indicates the end of input.

SAMPLE INPUT:

```
COST 1 1
1
34
15
COST 1 10
1
34
15
COST 10 1
1
34
15
COST 0 5
1
16
#
```

OUTPUT FORMAT

Each set of output data must begin with a single output line showing consisting of the word "`Set`", followed by a space (" "), and the current set number (counted from 1). This is followed by the cost of the binary digits for each house number, each cost being displayed as a decimal number on a separate line.

SAMPLE OUTPUT:

```
Set 1
1
6
4
Set 2
10
24
40
Set 3
1
42
4
Set 4
5
5
```

PROBLEM 5 – LOLLIES

Every day on his way home, little Billy passes by his great aunt Clara Mitchum's house. Generally he stops in for a chat with the great ACM (as he lovingly refers to her) and sometimes he asks for some lollies. When he does, she generally gives him some, but then adds "now don't be asking for any more for another N days" where N is some positive integer. If $N = 1$ that means he can ask for some on the next day, but for example if it is April 6 and $N = 4$ then he must wait until April 10 or later before asking for more lollies.

One day Billy happened to catch sight of the great ACM's calendar, and noted that each day was marked with two integers. He also noted that the first of these referred to the number of lollies the great ACM would give him on a particular day, and the second to the delay that would then be required before making another request. He copied down as much of the information as he could, and has passed it to you to analyse. His objective, of course, is to get as many lollies as he can.

Your task is to write a program which will report the total number of lollies that can be obtained by Billy, and provide a schedule for obtaining that amount. In the event that there are two or more ways to obtain the maximum number of lollies, Billy will choose the one where his first collection is as late as possible, and among all collections with that first date, his second collection is as late as possible, and so on.

INPUT FORMAT

The input text consists of a number of sets of unrelated problems. The first line of a set is a problem title consisting of a string of 1 to 20 letters. A single # on a line indicates the end of input.

The "title" line is followed by a sequence of "day" lines. Each problem set contains between 1 and 100 days, including the limits. In the given order, the first "day" line corresponds to day number 1, the second line to day number 2, the n-th line to day number n. Each "day" line consists of two integers separated by a single space:

- an integer L, which is the number of lollies available on that day ($1 \le L \le 100$),
- an integer N, which is the associated delay ($1 \le N \le 100$).

Conventionally, a delay N pointing to a day beyond the end of the current problem refers to a day with zero lollies and zero further delays ($L = 0$, $N = 0$).

SAMPLE INPUT:

```
January
1 1
2 2
3 3
February
10 3
7 1
5 2
1 1
March
2 3
1 1
3 7
2 7
#
```

OUTPUT FORMAT

Each report must follow the following format (use single spaces for spacing):

```
In <problem_title> <total_amount> <lollies> can be obtained:
On day <day_number> collect <day_amount> <lollies>.
On day <day_number> collect <day_amount> <lollies>.
…
```

In this notation, *<problem_title>* represents the actual problem title, *<total_amount>*, *<day_amount>*, and *<day_number>* are numbers with self-described meaning, and *<lollies>* stands for either "lolly" or "lollies", as required by the context (the singular and plural forms must be used appropriately). Days must be given in increasing sequence numbers. Each group report should be separated from the next by a blank line.

SAMPLE OUTPUT:

```
In January 4 lollies can be obtained:
On day 1 collect 1 lolly.
On day 3 collect 3 lollies.

In February 12 lollies can be obtained:
On day 2 collect 7 lollies.
On day 3 collect 5 lollies.

In March 4 lollies can be obtained:
On day 2 collect 1 lolly.
On day 3 collect 3 lollies.
```

PROBLEM 6 – MAZE MADNESS

You have been placed somewhere in a maze and you wish to escape by the shortest possible route. Fortunately you have been given a map of the maze. Before setting off, you wish to calculate the distance you need to travel. Your task is to write a program that will calculate the shortest distance to leave the maze. Note that there may be more than one exit and the specified start position could be at any location within the maze.

The maze is set on a grid that has M columns and N rows, with $1 \le M, N \le 100$. Some squares of this grid have impenetrable walls of negligible thickness between them (or on their outside border). You may move from any square to a horizontally or vertically adjacent square (possibly outside the maze, thus escaping) provided that there is no wall between them. Each single move between squares adds 1 meter to the distance travelled.

INPUT FORMAT

The input involves a series of scenarios. Within each of the scenarios the first line has the size of the maze. This is given as two numbers M and N. Then the maze is drawn on $2*N+1$ lines and $2*M+1$ columns using the printable characters "-", "|", "+", ".", " " (space), and "s":

- "|" is used for "vertical" walls,
- "-" is used for "horizontal" walls,
- "+" is used to indicate boundaries between rows and columns (there are always $(N+1)*(M+1)$ of these),
- "." is used for wall openings,
- " " (space) is used for empty squares,
- "s" is used to show your start location (there is exactly one "s").

A line with "0 0" indicates the end of the scenarios.

SAMPLE INPUT:

```
1 1
+-+
|s.
+-+
3 2
+-+-+.+
| .s| |
+-+.+-+
| . . .
+-+-+-+
5 6
+-+-+-+-+-+
| . . . . |
+-+-+.+-+-+
| |s. . . |
+.+-+-+-+.+
| | . . . |
+.+-+-+-+.+
| | . . . |
+.+-+.+-+.+
| . . | | |
+-+-+.+.+.+
. . . | . |
+-+-+-+-+-+
3 2
+-+-+.+
| .s| |
+-+-+-+
| . . .
+-+-+-+
0 0
```

OUTPUT FORMAT

Output a single line for each of the scenarios. This line should contain either "`Maze i: d`" or "`Maze i: No escape!`", where `i` is the scenario number (counting from 1) and `d` is the minimum distance (in meters) needed to escape (use single spaces for spacing).

SAMPLE OUTPUT:

```
Maze 1: 1
Maze 2: 3
Maze 3: 12
Maze 4: No escape!
```

PROBLEM 2 – BOOK PAGES

For the purposes of this problem, we will assume that every page in an Acmonian book is numbered sequentially, and that the first page is numbered 1.

How many digits would you need to use to number the pages of a 10 page book? Pages 1 to 9 would require 1 digit each (total 9), and page 10 would require 2 digits. This makes 11 digits. Similarly, a book of 34 pages would require 59 digits.

Can we work backwards? If you are told that a book requires 13 digits to number its pages, can you work out how many pages the book has? I hope so, because that is all you have to do for this problem. Each line in the input file represents the number of digits used in numbering a book. Your answer will be the number of pages the book has. If the number supplied cannot possibly be valid, your answer should be "Impossible!" Beware that Acmonian books can be quite large, and the number of digits required for a given Acmonian book can reach 2,000,000,000.

INPUT FORMAT

Each line in the input file contains a single integer, between 1 and 2,000,000,000, representing a number of digits used in numbering the pages of a book. A single `#` on a line indicates the end of input.

SAMPLE INPUT:

```
11
13
59
60
1999999998
#
```

OUTPUT FORMAT

Output for each input number must be on a single line. If the input value is valid, output the number of pages in the book. Otherwise, output "`Impossible!`"

SAMPLE OUTPUT:

```
10
11
34
Impossible!
234567900
```

PROBLEM 4 – COMPANY PARTIES

The Acme company has an hierarchical organization, i.e., a tree-like structure with the CEO at the root and each other employee a child node of his/her manager. In addition to his/her position in the organization, each employee has a unique employee id (a string with no particular meaning) and a sociability measure (an integer number).

The CEO of the Acme company wants to organize a party for their employees. To make the party agreeable the CEO wants to make invitations such that:

- the CEO attends the party,
- an employee can be invited only if his/her direct manager is absent,
- the sum of the sociability measures of all who attend is a maximum.

Write a program that will determine the maximum sociability sum under these conditions for a given company structure.

INPUT FORMAT

The input text consists of a number of company structures. The first line of a set is a title giving the company name. The company name may contain any printable non-space characters; and embedded spaces are also permitted. A single # on a line indicates the end of input.

The "name" line is followed by one line consisting of a single integer n, $1 \leq n \leq 100{,}000$, that indicates the number of employees in this company. This line is followed by n further lines, one line for each employee.

Each "employee" line consists of three items separated by single spaces:

- an employee id, which is a sequence of 1 to 10 letters and/or digits,
- a sociability measure, which is an integer number between 0 and 100,
- a manager id, which is the employee id of the current employee's direct manager, or the character "–" for the CEO.

The employee id is a unique identifier within the company. The order in which the employees appear is arbitrary, i.e., not related to their employee or manager ids.

SAMPLE INPUT:

```
ACME 1
1
ID0 10 -
ACME II
2
ID0 10 -
ID1 21 ID0
ACME, INC.
8
ID4 20 ID3
ID5 1 ID4
ID6 1 ID3
ID7 10 ID6
ID0 10 -
ID1 21 ID0
ID2 10 ID1
ID3 11 ID0
#
```

OUTPUT FORMAT

There is a single output line for each company. Each output line consists of the company title, followed by a colon and a space, and finally the maximum attainable sociability measure under the above conditions.

SAMPLE OUTPUT:

```
ACME 1: 10
ACME II: 10
ACME, INC.: 50
```

PROBLEM 9 – POLYGONAL LINES

A rectangle is to be cut by a sequence of one or more straight line segments joining a start node to an end node, both on the rectangle border.

Write a program that will read in details of the rectangle and the dividing line and determine whether the cut produces exactly two parts that could slide apart while remaining in the same plane.

The following sample diagrams contain:

1. several cases where the answer is "`Yes`", i.e., the cutting line produces exactly two parts that can slide away as required (**1** - **5**), and
2. several cases where the answer is "`No`", i.e., the cutting line doesn't produce exactly two parts as required (**6** - **10**).

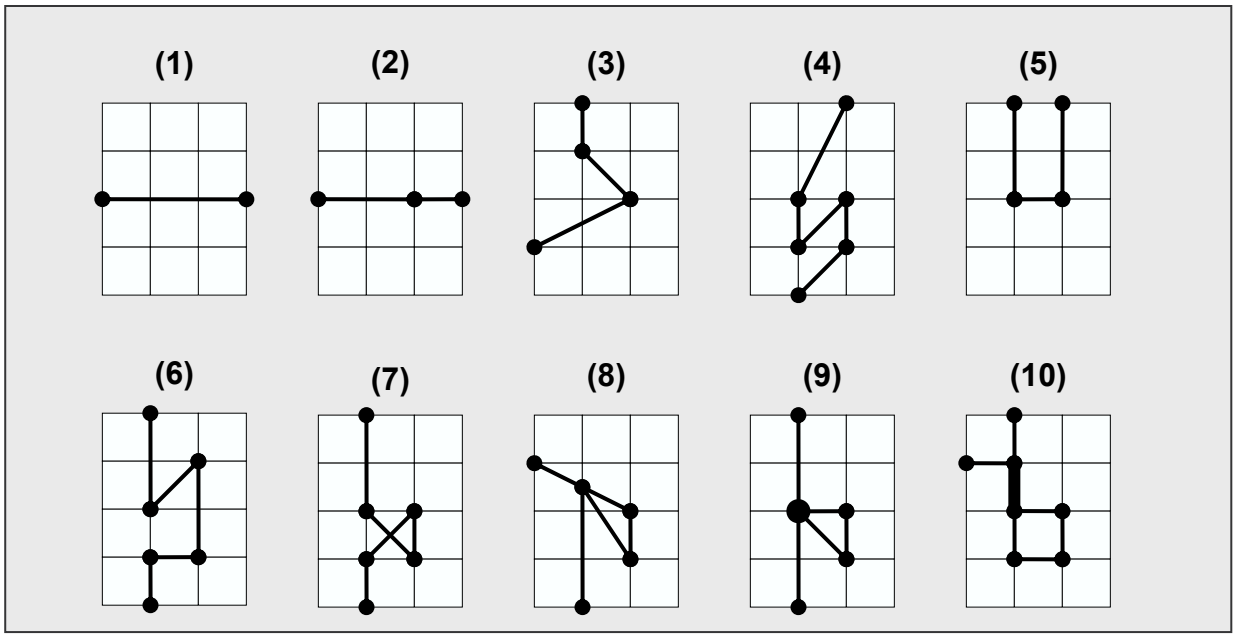

Diagram (**6**) shows two interlocked parts that cannot slide apart. Diagrams (**7** – **10**) show cuttings that produce three parts instead of the required two. Diagram (**7**) shows two intersecting segments. Diagram (**8**) shows a node that touches another segment. Diagram (**9**) shows two duplicate nodes that overlap (on the thicker spot). Diagram (**10**) shows two segments that overlap (along the thicker line). This overlapping will be more obvious in the sample input section below (where the grid size is assumed to be 10).

The programmer that initially received this task noticed that if the two parts can slide apart then they can always slide apart along the slope of at least one of the given line segments. However, he was unable to put this idea to work. Your task is to help him and write the required program.

INPUT FORMAT

Input consists of a number of scenarios. Each scenario starts with a "title" line containing a scenario title followed by three integers X_{MAX}, Y_{MAX}, and N, separated by single spaces. The scenario title is a string of 1 to 20 letters, digits, and underscores (with no intervening spaces). It is assumed that our rectangle is aligned with the axes with the origin at the bottom-left corner, and that X_{MAX} and Y_{MAX} specify the top-right corner, where $10 \le X_{MAX}, Y_{MAX} \le 1{,}000{,}000$. N represents the number of nodes of the cutting line, $2 \le N \le 200{,}000$. A single `#` on a line indicates the end of input.

This "title" line is followed by one or more lines, as needed to describe all nodes of the cutting line, in succession, 0, 1, 2, …, N-1. Each "nodes" line contains one or more pairs of non-negative integers, each giving the x and y coordinates of the corresponding node.

You can assume that:

- the start and the end nodes are distinct and lie on the rectangle borders,
- all other nodes lie within the interior area of the rectangle,
- there are no successive duplicate nodes (i.e., no 0 length segments),
- there are no successive overlapping segments (such as "`0 0 4 4 2 2`").

SAMPLE INPUT:

```
Case_1 30 40 2
0 20 30 20
Case_2 30 40 3
30 20 20 20 0 20
Case_3 30 40 4
0 10 20 20 10 30 10 40
Case_4 30 40 6
20 40 10 20 10 10
20 20 20 10 10 0
Case_5 30 40 4
20 40 20 20 10 20 10 40
Case_6 30 40 6
10 0 10 10 20 10 20 30 10 20 10 40
Case_7 30 40 6
10 0 10 10 20 20 20 10 10 20 10 40
Case_8 30 40 5
10 0 10 25 20 10 20 20 0 30
Case_9 30 40 6
10 0 10 20 20 10 20 20 10 20 10 40
Case_10 30 40 7
10 40 10 20 20 20 20 10 10 10 10 30 0 30
#
```

OUTPUT FORMAT

Output consists of one line for each scenario. There are two cases:

1. The cutting line meets all our requirements and produces exactly two parts that can slide apart. In this case assume that ((*x1*, *y1*), (*x2*, *y2*)) is the first of the line segments whose slope can be used for sliding apart the two parts. Output the scenario title, followed by a colon (“:”), a space (“ ”), the word “`Yes`”, another space (“ ”), and then the four integers *x1*, *y1*, *x2*, *y2*, separated by single spaces (these coordinates must appear in their input sequence).

2. Otherwise output the scenario title, followed by a colon (“:”), a space (“ ”), and the word “`No`”.

SAMPLE OUTPUT:

```
Case_1: Yes 0 20 30 20
Case_2: Yes 30 20 20 20
Case_3: Yes 0 10 20 20
Case_4: Yes 10 10 20 20
Case_5: Yes 20 40 20 20
Case_6: No
Case_7: No
Case_8: No
Case_9: No
Case_10: No
```

PROBLEM 3 – VOWELS FREQUENCIES

The English alphabet consists of 26 letters. Five of these (`a`, `e`, `i`, `o` and `u`) are classified as vowels, the remaining 21 as consonants. Almost every English word contains at least one vowel ("rhythm" is one of the few exceptions).

In this problem you will be given a number of pieces of English text. Your task is to determine the frequency of each vowel that is found in the piece, and to display the answers sorted by frequency, highest frequency first. Where two vowels are equally frequent, they are to be displayed in alphabetical order.

As you can see from the examples below, upper case and lower case letters are considered to be the same letter in this problem. Use lower case in your output. As you can see from the second example, a frequency of zero must still be displayed.

INPUT FORMAT

Each piece of text to be analysed is on a separate line of the input file. Each line has at most 200 characters. A single `#` on a line indicates the end of input.

SAMPLE INPUT:

```
This piece of text was written in the city of Auckland.
ACM Programming Contest.
#
```

OUTPUT FORMAT

Output for a problem must be on a single line. Each vowel must be output in lower case, followed by a colon, followed by the frequency of that vowel. There must be one space before the next letter, and a dot at the end.

SAMPLE OUTPUT:

```
e:5 i:5 a:3 o:2 u:1.
a:2 o:2 e:1 i:1 u:0.
```