Archives of the Sphere Online Judge

partial problemset

Editors:

Nikola P Borisov
Damon Doucet
Trân Ha?i Đăng
Zhang Zhiyong
Melvin
Mir Wasi Ahmed
Infinity
Thanh-Vy Hua
yk .o0o.
sieunhan
Felix Halim
Ngô Minh Đu+'c
T-7
Vu+o+ng Trung Hie^'u
Nghia

Last updated: 2011-05-22 19:15:10

Preface

This electronic material contains a set of algorithmic problems, forming the archives of the Sphere Online Judge (http://www.spoj.pl/), partial problemset. The document can be accessed at the following URLs:

- in PostScript format: http://www.spoj.pl/problems/partial.ps
- in Portable Document Format: http://www.spoj.pl/problems/partial.pdf

These resources are constantly updated to synchronise with the ever-changing hypertext version of the problems, and to include newly added problems. If you have obtained this document from another source, it is strongly recommended that you should download the current version from one of the aforementioned URLs.

Enjoy problem-solving at the Sphere Online Judge!

Disclaimer from the Editors. Despite our best efforts, it is possible that this document contains errors or that some of the content differs slightly from its original hypertext form. We take no responsibility for any such faults and their consequences. We neither authorise nor approve use of this material for any purpose other than facilitating problem solving at the Sphere Online Judge site; nor do we guarantee its fitness for any purpose whatsoever.

The layout of the problems in this document is the copyright of the Editors named on the cover (as determined by the appropriate footers in the problem description). The content is the copyright of the respective Editor unless the copyright holder is otherwise stated in the 'resource' section. The document as a whole is not protected by copyright, and fragments of it are to be regarded independently. No responsibility is taken by the Editors if use or redistribution of this document violates either their or third party copyright laws. When referring to or citing the whole or a fragment of this document, please state clearly the aforementioned URLs at which the document is to be found, as well as the resources from which the problems you are referring to originally came.

Remarks concerning this document should be sent to the following e-mail address: contact@spoj.pl.

Table of Contents

SPOJ Problem Set (partial)

1486. The Ants in a Tree

Problem code: PT07I

In Amber's childhood, he usually liked to observe some little things for tickling his little curiosity. He often found it interesting to climb up a tree, sit on a branch and watch the movement of a group of lovely ants on the branches of the tree.

Amber finds there are *n* ant holes and *m* ants in the tree now. Because of his careful observation, he knows all ants' behaviors, the *i*-th ant wanna travel from one hole s_i to another hole t_i at the speed v_i.

During the ant's travel, if two ants arrive at the same position (meet or chase), they will touch their feelers for exchanging the information about food or danger. Even at the moment that the travel starts or finishes, the ant can also touch other's feelers. But after the travel finishes, the ant will enter into the hole and never touch feelers. What amber wonders is to count the times of touchings during the whole traveling process.

Consider there are *n* - 1 branches in the tree. Each branch connects the adjacent ant holes and has a particular length. Assume there is always a path that consists of branches between any two ant holes. Assume that no two ants have the same speeds and the touching doesn't cost any time.

Input

Input consists of multiple testcases. The first line contains one integer *t* (1 <= *t* <= 20). For each testcase, the input format is following.

The first line contains one integer *n* (1 <= *n* <= 10^6). In next *n* - 1 line, the *i*-th line contains an integer triple (u_i, v_i, w_i) (1 <= u_i, v_i <= *n*, u_i != v_i, 1 <= w_i <= 10^3). The triple means there is a branch with the length w_i between node u_i and node v_i.

The next line contains one integer *m* (1 <= *m* <= 10^3). In next *m* line, the *i*-th line contains an integer triple (s_i, t_i, v_i) (1 <= s_i, t_i <= *n*, 1 <= v_i <= 10^6). The triple means that the *i*-th ant's travel is from s_i to t_i at the speed v_i.

Output

For each testcase, print a line that consists of an integer that means the times of the feeler touchings.

Example

```
Input:
1
3
1 2 1
2 3 1
3
1 3 1
3 1 1
```

```
1 2 3
```

Output:

```
2
```

Note: It’s a partly correct problem, so score is set for each case. When you pass all cases, you’ll get 300 points

---

Added by:	Thanh-Vy Hua
Date:	2007-04-07
Time limit:	8s-15s
Source limit:	50000B
Languages:	All
Resource:	Co-author Amber

SPOJ Problem Set ()

3077. Copier Reduction

Problem code: REDUCT

What do you do if you need to copy a 560x400mm image onto a standard sheet of US letter-size paper (which is about 216x280mm), while keeping the image as large as possible? You can rotate the image 90 degrees (so that it is in landscape mode), then reduce it to 50% of its original size so that it is 200x280mm.Then it will fit on the paper without overlapping any edges. Your job is to solve this problem in general.

Input

The input consists of one or more test cases, each of which is a single line containing four positive integers *A*, *B*, *C*, and *D*, separated by a space, representing an *A*x*B*mm image and a *C*x*D*mm piece of paper. All inputs will be less than one thousand. Following the test cases is a line containing four zeros that signals the end of the input.

Output

For each test case, if the image fits on the sheet of paper without changing its size (but rotating it if necessary), then the output is 100%. If the image must be reduced in order to fit, the output is the largest *integer* percentage of its original size that will fit (rotating it if necessary). Output the percentage exactly as shown in the examples below. You can assume that no image will need to be reduced to less than 1% of its original size, so the answer will always be an integer percentage between 1% and 100%, inclusive.

Example

```
Input:
560 400 218 280
10 25 88 10
8 13 5 1
9 13 10 6
199 333 40 2
75 90 218 280
999 99 1 10
0 0 0 0

Output:
50%
100%
12%
66%
1%
100%
1%
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2005

SPOJ Problem Set ()

3079. Consecutive Digits

Problem code: CONDIG

As a recruiting ploy, Google once posted billboards in Harvard Square and in the Silicon Valley area just stating "{first 10-digit prime found in consecutive digits of e}.com". In other words, find that 10-digit sequence and then connect to the web site -- and find out that Google is trying to hire people who can solve a particular kind of problem.

Not to be outdone, Gaggle (a loosy-goosy fuzzy logic search firm), has devised its own recruiting problem. Consider the *base 7* expansion of a rational number. For example, the first few digits of the base 7 expansion of $1/5_{10} = 0.12541..._7$, $33/4_{10} = 11.15151..._7$, and $6/49_{10} = 0.06000..._7$, From this expansion, find the digits in a particular range of positions to the right of the "decimal" point.

Input

The input file begins with a line containing a single integer specifying the number of problem sets in the file. Each problem set is specified by four base 10 numbers on a single line, n d b e, where n and d are the numerator and denominator of the rational number and 0 <= n <= 5,000 and 1 <= d <= 5,000. b and e are the beginning and ending positions for the desired range of digits, with 0 <= b,e <= 250 and 0 <= (e-b) <= 20. Note that 0 is the position immediately to the right of the decimal point.

Output

Each problem set will be numbered (beginning at one) and will generate a single line: Problem set k: n / d, base 7 digits b through e: result where k is replaced by the problem set number, result is your computed result, and the other values are the corresponding input values.

Example

```
Input:
4
1 5 0 0
6 49 1 3
33 4 2 7
511 977 122 126

Output:
Problem set 1: 1 / 5, base 7 digits 0 through 0: 1
Problem set 2: 6 / 49, base 7 digits 1 through 3: 600
Problem set 3: 33 / 4, base 7 digits 2 through 7: 151515
Problem set 4: 511 / 977, base 7 digits 122 through 126: 12425
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2005

SPOJ Problem Set (partial)

3080. Painter

Problem code: PAINTER

The local toy store sells small fingerpainting kits with between three and twelve 50ml bottles of paint, each a different color. The paints are bright and fun to work with, and have the useful property that if you mix *X* ml each of any three different colors, you get *X* ml of gray. (The paints are thick and "airy", almost like cake frosting, and when you mix them together the volume doesn't increase, the paint just gets more dense.) None of the individual colors are gray; the only way to get gray is by mixing exactly three distinct colors, but it doesn't matter which three. Your friend Emily is an elementary school teacher and every Friday she does a fingerpainting project with her class. Given the number of different colors needed, the amount of each color, and the amount of gray, your job is to calculate the number of kits needed for her class.

Input

The input consists of one or more test cases, followed by a line containing only zero that signals the end of the input. Each test case consists of a single line of five or more integers, which are separated by a space. The first integer *N* is the number of different colors (3 <= *N* <= 12). Following that are *N* different nonnegative integers, each at most 1,000, that specify the amount of each color needed. Last is a nonnegative integer *G* <= 1,000 that specifies the amount of gray needed. All quantities are in ml.

Output

For each test case, output the smallest number of fingerpainting kits sufficient to provide the required amounts of all the colors and gray. Note that all grays are considered equal, so in order to find the minimum number of kits for a test case you may need to make grays using different combinations of three distinct colors.

Example

```
Input:
3 40 95 21 0
7 25 60 400 250 0 60 0 500
4 90 95 75 95 10
4 90 95 75 95 11
5 0 0 0 0 0 333
0


Output:
2
8
2
3
4
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2005

SPOJ Problem Set (partial)

3082. Primary X-Subfactor Series

Problem code: PRIMESUB

Let *n* be any positive integer. A *factor* of *n* is any number that divides evenly into *n*, without leaving a remainder. For example, 13 is a factor of 52, since 52/13 = 4. A *subsequence* of *n* is a number without a leading zero that can be obtained from *n* by discarding one or more of its digits. For example, 2, 13, 801, 882, and 1324 are subsequences of 8013824, but 214 is not (you can't rearrange digits), 8334 is not (you can't have more occurrences of a digit than appear in the original number), 8013824 is not (you must discard at least one digit), and 01 is not (you can't have a leading zero). A *subfactor* of *n* is an integer greater than 1 that is both a factor and a subsequence of *n*. 8013824 has subfactors 8, 13, and 14. Some numbers do not have a subfactor; for example, 6341 is not divisible by 6, 3, 4, 63, 64, 61, 34, 31, 41, 634, 631, 641, or 341.

An *x-subfactor series* of *n* is a decreasing series of integers n_1, ..., n_k, in which (1) $n = n_1$, (2) $k >= 1$, (3) for all $1 <= i < k$, n_{i+1} is obtained from n_i by first discarding the digits of a subfactor of n_i, and then discarding leading zeros, if any, and (4) n_k has no subfactor. The term "x-subfactor" is meant to suggest that a subfactor gets x'ed, or discarded, as you go from one number to the next. For example, 2004 has two distinct x-subfactor series, the second of which can be obtained in two distinct ways. The highlighted digits show the subfactor that was removed to produce the next number in the series.

2004 4
200***4*** ***20***0 0
200***4*** ***2***00 0

The *primary* x-subfactor series has maximal length (the largest *k* possible, using the notation above). If there are two or more maximal-length series, then the one with the smallest second number is primary; if all maximal-length series have the same first and second numbers, then the one with the smallest third number is primary; and so on. Every positive integer has a unique primary x-subfactor series, although it may be possible to obtain it in more than one way, as is the case with 2004.

Input

The input consists of one or more positive integers, each less than one billion, without leading zeroes, and on a line by itself. Following is a line containing only "0" that signals the end of the input.

Output

For each positive integer, output its primary x-subfactor series using the exact format shown in the examples below.

Example

Input:
```
123456789
7
2004
6341
8013824
0
```

Output:
```
123456789 12345678 1245678 124568 12456 1245 124 12 1
7
2004 200 0
6341
8013824 13824 1324 132 12 1
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2005

SPOJ Problem Set (partial)

3088. Packing

Problem code: PACK1

In the future the delivery services will be fully automated. A robot will come to your home to pick the boxes and leaves them in the central processing office where boxes for the same address are packed together. There is a machine that can pack two boxes into a one new box containing the privious two. If we want N boxes delivered to a certain city then this machine with N-1 operation will be able to consolidate them into single box.

Each box has its size and a price for packing it equal to this size. The size of the box resulting from the machine packing two boxes together is simply equal to the sum of the two boxes that are packed together. Your goal is to find out the minimum price for packing N boxes into a single one using the packing machine.

Input

On the first line there will be one number N (1 < N < 5000001) - the number of boxes. N lines follow each line with one number representing the size of N-th box. The size will be less then 1 000 000. In 50% of the test cases the size will be less then 4000.

Output

Your program should output a single integer - the minum price that have to be paid for packing the N boxes into a single one using N-1 operations of the machine.

Example

Input:
```
4
1
1
1
1
```

Output:
```
8
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	1s-25s
Source limit:	50000B
Languages:	All
Resource:	Bulgarian National Olympiad Selection Contest 4 2005

SPOJ Problem Set (partial)

3095. Symmetric Order

Problem code: SYMORD

In your job at Albatross Circus Management (yes, it's run by a bunch of clowns), you have just finished writing a program whose output is a list of names in nondescending order by length (so that each name is at least as long as the one preceding it). However, your boss does not like the way the output looks, and instead wants the output to appear more symmetric, with the shorter strings at the top and bottom and the longer strings in the middle. His rule is that each pair of names belongs on opposite ends of the list, and the first name in the pair is always in the top part of the list. In the first example set below, Bo and Pat are the first pair, Jean and Kevin the second pair, etc.

Input

The input consists of one or more sets of strings, followed by a final line containing only the value 0. Each set starts with a line containing an integer, *n*, which is the number of strings in the set, followed by *n* strings, one per line, sorted in nondescending order by length. None of the strings contain spaces. There is at least one and no more than 15 strings per set. Each string is at most 25 characters long.

Output

For each input set print "SET n" on a line, where n starts at 1, followed by the output set as shown in the sample output.

Example

Input:
```
7
Bo
Pat
Jean
Kevin
Claude
William
Marybeth
6
Jim
Ben
Zoe
Joey
Frederick
Annabelle
5
John
Bill
Fran
Stan
Cece
0
```

Output:

```
SET 1
Bo
Jean
Claude
Marybeth
William
Kevin
Pat
SET 2
Jim
Zoe
Frederick
Annabelle
Joey
Ben
SET 3
John
Fran
Cece
Stan
Bill
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2004

SPOJ Problem Set (partial)

3096. Overflowing Bookshelf

Problem code: BOOKSH

Agnes C. Mulligan is a fanatical bibliophile - she is constantly buying new books, and trying to find space for those books. In particular, she has a shelf for her "to be read" books, where she puts her newest books. When she decides to read one of these books, she removes it from the shelf, making space for more books. Sometimes, however, she buys a new book and puts it on the shelf, but because of limited space, this pushes one or more books off the shelf at the other end. She always adds books on the left side of the shelf, making books fall off the right side. Of course, she can remove a book from any location on the shelf when she wants to read one.

Your task will be to write a simulator that will keep track of books added and removed from a shelf. At the end of the simulation, display the books remaining on the shelf, in order from left to right. Books in each simulation will be identified by a unique, positive integer, 0 < I <= 100. There are three types of events in the simulation:

- Add: A new book is pushed on the left end of the shelf, pushing other books to the right as needed. No book moves to the right unless it is pushed by an adjacent (touching) book on its left. Any book that is not entirely on the shelf falls off the right edge. No single book will ever be wider than the given shelf. No book that is currently on the shelf will be added again.
- Remove: If the book is on the shelf, then the book is removed from the shelf, leaving a hole. If the book isn't on the shelf, the operation is ignored.
- End: End the simulation for this case and print the contents of the shelf.

Input

The input file will contain data for one or more simulations. The end of the input is signalled by a line containing -1. Each simulation will begin with the integer width of the shelf, s, 5 <= s <= 100, followed by a series of add and remove events. An add event is a single line beginning with an upper case 'A' followed by the book ID, followed by the integer width of the book, w, 0 < w <= s. A remove event is a single line beginning with an upper case 'R' followed by the the book ID. Finally, the end event is a line containing only a single upper case 'E'. Each number in an event is preceded by a single blank.

Output

For each simulation case, print a single line containing a label (as shown in the output sample), followed by the list of IDs of books remaining on the shelf, in order from left to right.

Input:
```
10
R 3
A 6 5
A 42 3
A 3 5
A 16 2
A 15 1
```

```
R 16
E
7
A 49 6
A 48 2
R 48
E
5
A 1 1
A 2 1
A 3 1
R 2
A 4 1
A 5 1
R 5
R 4
A 6 1
A 7 4
E
-1
```

Output:

```
PROBLEM 1: 15 3
PROBLEM 2:
PROBLEM 3: 7 6
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2005

SPOJ Problem Set (partial)

3097. Flow Layout

Problem code: FLOWLAY

A flow layout manager takes rectangular objects and places them in a rectangular window from left to right. If there isn't enough room in one row for an object, it is placed completely below all the objects in the first row at the left edge, where the order continues from left to right again. Given a set of rectangular dimensions and a maximum window width, you are to write a program that computes the dimensions of the final window after all the rectangles have been placed in it.

For example, given a window that can be at most 35 units wide, and three rectangles with dimensions 10 x 5, 20 x 12, and 8 x 13, the flow layout manager would create a window that looked like the figures below after each rectangle was added.

insert 10x5 rectangle insert 20x12 rectangle insert 8x13 rectangle

The final dimensions of the resulting window are 30 x 25, since the width of the first row is 10+20 = 30 and the combined height of the first and second rows is 12+13 = 25.

Input

The input consists of one or more sets of data, followed by a final line containing only the value 0. Each data set starts with a line containing an integer, *m*, 1 <= m <= 80, which is the maximum width of the resulting window. This is followed by at least one and at most 15 lines, each containing the dimensions of one rectangle, width first, then height. The end of the list of rectangles is signaled by the pair -1 -1, which is not counted as the dimensions of an actual rectangle. Each rectangle is between 1 and 80 units wide (inclusive) and between 1 and 100 units high (inclusive).

Output

For each input set print the width of the resulting window, followed by a space, then the lowercase letter "x", followed by a space, then the height of the resulting window.

Example

```
Input:
35
10 5
20 12
8 13
-1 -1
25
10 5
20 13
3 12
-1 -1
15
5 17
5 17
```

```
5 17
7 9
7 20
2 10
-1 -1
0
```

Output:

```
30 x 25
23 x 18
15 x 47
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Mid-Central Regional Contest 2004

SPOJ Problem Set (partial)

3098. Tree Grafting

Problem code: GRAFT

Trees have many applications in computer science. Perhaps the most commonly used trees are rooted binary trees, but there are other types of rooted trees that may be useful as well. One example is ordered trees, in which the subtrees for any given node are ordered. The number of children of each node is variable, and there is no limit on the number. Formally, an ordered tree consists of a finite set of nodes T such that

- there is one node designated as the root, denoted root(T);
- the remaining nodes are partitioned into subsets T1, T2, ..., Tm, each of which is also a tree (subtrees).

Also, define root(T1), ..., root(Tm) to be the children of root(T), with root(Ti) being the i-th child. The nodes root(T1), ..., root(Tm) are siblings.

It is often more convenient to represent an ordered tree as a rooted binary tree, so that each node can be stored in the same amount of memory. The conversion is performed by the following steps:

1. remove all edges from each node to its children;
2. for each node, add an edge to its first child in T (if any) as the left child;
3. for each node, add an edge to its next sibling in T (if any) as the right child.

This is illustrated by the following:

```
       0                          0
     / | \                       /
    1  2  3        ===>         1
      / \                        \
     4   5                        2
                                 / \
                                4   3
                                 \
                                  5
```

In most cases, the height of the tree (the number of edges in the longest root-to-leaf path) increases after the conversion. This is undesirable because the complexity of many algorithms on trees depends on its height.

You are asked to write a program that computes the height of the tree before and after the conversion.

Input

The input is given by a number of lines giving the directions taken in a depth-first traversal of the trees. There is one line for each tree. For example, the tree above would give dudduduudu, meaning 0 down to 1, 1 up to 0, 0 down to 2, etc. The input is terminated by a line whose first character is #. You may assume that each tree has at least 2 and no more than 10000 nodes.

Output

For each tree, print the heights of the tree before and after the conversion specified above. Use the format:

```
  Tree t: h1 => h2
```

where t is the case number (starting from 1), h1 is the height of the tree before the conversion, and h2 is the height of the tree after the conversion.

Example

Input:
```
dudduduudu
ddddduuuuu
ddddduduuuu
dddduuduuu
#
```

Output:
```
Tree 1: 2 => 4
Tree 2: 5 => 5
Tree 3: 4 => 5
Tree 4: 4 => 4
```

Added by:	Nikola P Borisov
Date:	2008-10-01
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Rocky Mountain Regional Contest 2007

SPOJ Problem Set (partial)

3132. Server Relocation

Problem code: UPTIME

Michael has a powerful computer server that has hundreds of parallel processors and terabytes of main memory and disk space. Many important computations run continuously on this server, and power must be supplied to the server without interruption.

Michael's server must be moved to accommodate new servers that have been purchased recently. Fortunately, Michael's server has two redundant power supplies---as long as at least one of the two power supplies is connected to an electrical outlet, the server can continue to run. When the server is connected to an electrical outlet, it can be moved to any location which is not further away from the outlet than the length of the cord used to connect to the outlet.

Given which outlet Michael's server is plugged into initially and finally, and the locations of outlets in the server room, you should determine the smallest number of times you need to plug a cord into an electrical outlet in order to move the server while keeping the server running at all times. Note that, in the initial and final configuration, only one cord is connected to the power outlet.

Input

The first line of input is an integer giving the number of cases to follow. For each case, the first line is of the form

OUTLETS OUTLET_INITIAL OUTLET_FINAL LENGTH1 LENGTH2

where

- OUTLETS is the number of outlets in the server room (2 <= OUTLETS <= 1000).
- OUTLET_INITIAL is the index (starting from 1) of the outlet the server is initially connected to.
- OUTLET_FINAL is the index (starting from 1) of the outlet the server is finally connected to.
- LENGTH1 and LENGTH2 are the positive lengths of the two power cords, with at most three digits of precision after the decimal point (0 < LENGTH1, LENGTH2 <= 30000).

These are followed by OUTLETS lines giving the integer coordinates of the wall outlets, one per line, with the k-th line giving the location of the k-th outlet. All coordinates are specified as two integers (x- and y-coordinates) separated by a space, with absolute values at most 30000. You may assume that all coordinates are distinct, and that the initial outlet and the final outlet are different.

Output

For each case, print the minimum number of times you need to plug a cord into an electrical outlet in order to move the server to the final location while keeping the server running at all times. If this is not possible, print "Impossible".

Example

Input:
```
2
4 1 4 2.000 1.000
0 0
0 4
4 0
4 4
9 1 4 2.000 3.000
0 7
-6 2
-3 3
6 2
-6 -3
3 -3
6 -3
-3 -7
0 -7
```

Output:
```
Impossible
8
```

Added by:	Nikola P Borisov
Date:	2008-10-11
Time limit:	5s-15s
Source limit:	50000B
Languages:	All
Resource:	Rocky Mountain Regional ACM-ICPC Contest 2007

SPOJ Problem Set (partial)

3185. Linear Pachinko

Problem code: LINEARPA

This problem is inspired by Pachinko, a popular game in Japan. A traditional Pachinko machine is a cross between a vertical pinball machine and a slot machine. The player launches small steel balls to the top of the machine using a plunger as in pinball. A ball drops through a maze of pins that deflect the ball, and eventually the ball either exits at a hole in the bottom and is lost, or lands in one of many gates scattered throughout the machine which reward the player with more balls in varying amounts. Players who collect enough balls can trade them in for prizes.

For the purposes of this problem, a linear Pachinko machine is a sequence of one or more of the following: holes (``."), floor tiles (``_"), walls (``|"), and mountains (``/\"). A wall or mountain will never be adjacent to another wall or mountain. To play the game, a ball is dropped at random over some character within a machine. A ball dropped into a hole falls through. A ball dropped onto a floor tile stops immediately. A ball dropped onto the left side of a mountain rolls to the left across any number of consecutive floor tiles until it falls into a hole, falls off the left end of the machine, or stops by hitting a wall or mountain. A ball dropped onto the right side of a mountain behaves similarly. A ball dropped onto a wall behaves as if it were dropped onto the left or right side of a mountain, with a 50% chance for each. If a ball is dropped at random over the machine, with all starting positions being equally likely, what is the probability that the ball will fall either through a hole or off an end? For example, consider the following machine, where the numbers just indicate character positions and are not part of the machine itself:

```
123456789
/\.|__/\.
```

The probabilities that a ball will fall through a hole or off the end of the machine are as follows, by position: 1 = 100%, 2 = 100%, 3 = 100%, 4 = 50%, 5 = 0%, 6 = 0%, 7 = 0%, 8 = 100%, 9 = 100%. The combined probability for the whole machine is just the average, which is approximately 61.111%.

Input

The input consists of one or more linear Pachinko machines, each 1âEuro"79 characters long and on a line by itself, followed by a line containing only "#" that signals the end of the input.

Output

For each machine, compute as accurately as possible the probability that a ball will fall through a hole or off the end when dropped at random, then output a single line containing that percentage truncated to an integer by dropping any fractional part.

Example

Input:
```
/\.|__/\.
_._/\_|.__/\./\_
...
___
./\.
_/\_
_|.|_|.|_|.|_
____|_____
#
```

Output:
```
61
53
100
0
100
50
53
10
```

Added by: Nikola P Borisov
Date: 2008-10-20
Time limit: 5s
Source limit:50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2006

SPOJ Problem Set (partial)

3186. Frugal Search

Problem code: FRSEARCH

For this problem you will write a search engine that takes a query, searches a collection of words, and finds the lexicographically smallest word that matches the query (i.e., the matching word that would appear first in an English dictionary). A query is a sequence of one or more terms separated by single vertical bars (``|"). A term is one or more letters followed by zero or more signed letters. A signed letter is either +*s* (‘positive" *s*) or −*s* (``negative" *s*), where *s* is a single letter. All letters are lowercase, and no letter will appear more than once within a term. A query will not contain spaces. A term matches a word if the word contains at least one of the unsigned letters, all of the positive letters, and none of the negative letters; a query matches a word if at least one of its terms matches the word.

Input

The input consists of one or more test cases followed by a line containing only ``#" that signals the end of the input. Each test case consists of 1âEuro"100 words, each on a line by itself, followed by a line containing only ``*" that marks the end of the word list, followed by one or more queries, each on a line by itself, followed by a line containing only ``* *" that marks the end of the test case. Each word will consist of 1-20 lowercase letters. All words within a test case will be unique. Each query will be as defined above and will be 1-79 characters long.

Output

For each query, output a single line containing the lexicographically smallest word within that test case that matches the query, or the word `NONE` if there is no matching word. At the end of each test case, output a dollar sign on a line by itself.

Example

```
Input:
elk
cow
bat
*
ea
acm+e
nm+o|jk+l
**
debian
slackware
gentoo
ubuntu
suse
fedora
mepis
*
yts
cab-e+n
r-e|zjq|i+t|vs-p+e-u-c
```

```
**
#
```

Output:

```
bat
NONE
elk
$
gentoo
ubuntu
NONE
$
```

Added by:	Nikola P Borisov
Date:	2008-10-20
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	Mid-Central Regional ACM-ICPC Contest 2006

SPOJ Problem Set (partial)

3187. Go Go Gorelians

Problem code: GOGOGORE

The Gorelians travel through space using warp links. Travel through a warp link is instantaneous, but for safety reasons, an individual can only warp once every 10 hours. Also, the cost of creating a warp link increases directly with the linear distance between the link endpoints.

The Gorelians, being the dominant force in the known universe, are often bored, so they frequently conquer new regions of space in the following manner.

1. The initial invasion force finds a suitable planet and conquers it, establishing a Regional Gorelian Galactic Government, hereafter referred to as the RGGG, that will govern all Gorelian matters in this region of space.
2. When the next planet is conquered, a single warp link is constructed between the new planet and the RGGG planet. Planets connected via warp links in this manner are said to be part of the Regional Gorelian Planetary Network, that is, the RGPN.
3. As additional planets are conquered, each new planet is connected with a single warp link to the nearest planet already in the RGPN, thus keeping the cost of connecting new planets to the network to a minimum. If two or more planets are equidistant from the new planet, the new planet is connected to whichever of them was conquered first.

This causes a problem however. Since planets are conquered in a more-or-less random fashion, after a while, the RGGG will probably not be in an ideal location. Some Gorelians needing to consult with the RGGG might only have to make one or two warps, but others might require dozens--very inconvenient when one considers the 10-hour waiting period between warps.

So, once each Gorelian year, the RGGG analyzes the RGPN and relocates itself to an optimal location. The optimal location is defined as a planet that minimizes the maximum number of warps required to reach the RGGG from any planet in the RGPN. As it turns out, there is always exactly one or two such planets. When there are two, they are always directly adjacent via a warp link, and the RGGG divides itself evenly between the two planets.

Your job is to write a program that finds the optimal planets for the RGGG. For the purposes of this problem, the region of space conquered by the Gorelians is defined as a cube that ranges from (0,0,0) to (1000,1000,1000).

Input

The input consists of a set of scenarios where the Gorelians conquer a region of space. Each scenario is independent. The first line of the scenario is an integer *N* that specifies the total number of planets conquered by the Gorelians. The next *N* lines of the input specify, in the order conquered, the *ID*s and coordinates of the conquered planets to be added to the RGPN, in the format *IDXYZ*. An *ID* is an integer from 1 to 1000. *X*, *Y*, and *Z* are integers from 0 to 1000. A single space separates the numbers. A value of *N* = 0 marks the end of the input.

Output

For each input scenario, output the *ID*s of the optimal planet or planets where the RGGG should relocate. For a single planet, simply output the planet *ID*. For two planets, output the planet *ID*s, smallest *ID* first, separated by a single space.

Example

Input:
```
5
1 0 0 0
2 0 0 1
3 0 0 2
4 0 0 3
5 0 0 4
5
1 0 0 0
2 1 1 0
3 3 2 0
4 2 1 0
5 3 0 0
10
21 71 76 4
97 32 5 69
70 33 19 35
3 79 81 8
31 91 17 67
52 31 48 75
48 90 14 4
41 73 2 21
83 74 41 69
26 32 30 24
0
```

Output:
```
3
2 4
31 97
```

Added by:	Nikola P Borisov
Date:	2008-10-20
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	Mid-Central Regional ACM-ICPC Contest 2006

3188. Minimum Spanning Tree

Problem code: MST

Find the minimum spanning tree of the graph.

Input

On the first line there will be two integers N - the number of nodes and M - the number of edges. (1 <= N <= 10000), (1 <= M <= 100000)
> M lines follow with three integers i j k on each line representing an edge between node i and j with weight k. The IDs of the nodes are between 1 and n inclusive. The weight of each edge will be <= 1000000.

Output<h3>

Single number representing the total weight of the minimum spanning tree on this graph. There will be only one possible MST.

Example

```
Input:
4 5
1 2 10
2 3 15
1 3 5
4 2 2
4 3 40

Output:
17
```

Added by:	**Nikola P Borisov**
Date:	**2008-10-20**
Time limit:	**5s-10s**
Source limit:	**50000B**
Languages:	**All**

SPOJ Problem Set (partial)

3448. Connect

Problem code: CONNECT2

Your task is to decide if a specified sequence of moves in the board game Connect ends with a winning move.

\epsfbox{p3381.eps}

In this version of the game, different board sizes may be specified. Pegs are placed on a board at integer coordinates in the range [0, *N*]. Players Black and White use pegs of their own color. Black always starts and then alternates with White, placing a peg at one unoccupied position (*x*, *y*) . Black's endzones are where *x* equals 0 or *N* , and White's endzones are where *y* equals 0 or *N* . Neither player may place a peg in the other player's endzones. After each play, the latest position is connected by a segment to every position with a peg of the same color that is a chess knight's move away (2 away in one coordinate and 1 away in the other), provided that a new segment will touch no segment already added, except at an endpoint. Play stops after a winning move, which is when a player's segments complete a connected path between the player's endzones.

For example, Figure 1 shows a board with *N* = 4 after the moves (0,2), (2,4), and (4,2). Figure 2 adds the next move (3,2). Figure 3a shows a poor next move of Black to (2,3). Figure 3b shows an alternate move for Black to (2,1) which would win the game.

Figure 4 shows the board with *N* = 7 after Black wins in 11 moves:

(0, 3), (6, 5), (3, 2), (5, 7), (7, 2), (4, 4), (5, 3), (5, 2), (4, 5), (4, 0), (2, 4)

Input

The input contains from 1 to 20 datasets followed by a line containing only two zeroes, '`0 0`'. The first line of each dataset contains the maximum coordinate *N* ($3 < N < 21$) and the number of total moves, *M* ($4 < M < 250$) , with *M* odd, so Black will always be the last player. The dataset ends with one or more lines each containing two or more coordinate pairs, with a total of *M* coordinate pairs. All numbers on any line will be separated by blanks. All data will be legal. There will never be a winning move before the last move.

Output

The output contains one line for each data set: '`yes`' if the last move is a winning move and '`no`' otherwise.

Example

```
Input:
4 5
0 2 2 4 4 2 3 2 2 3
4 5
0 2 2 4 4 2 3 2 2 1
```

```
7 11
0 3 6 5 3 2 5 7 7 2 4 4
5 3 5 2 4 5 4 0 2 4
0 0
```

Output:

```
no
yes
yes
```

Added by:	Nikola P Borisov
Date:	2008-12-04
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	ICPC North America Pacific Northwest Region

SPOJ Problem Set (partial)

3449. Sum of Squares

Problem code: SUMSQ

We are interested in how many different sequences of N non negative integers there are that have the sum of their squares less than S. Note that the squence (1, 2) is different from the sequence (2, 1).

Input

The input consists of only one line with two integers N (0 < N < 30) and S (S < 100).

Output

A single integer representing the number of different sequences that have the sum of their squares less than S.

Example

```
Input:
1 4

Output:
2
```

Added by:	Nikola P Borisov
Date:	2008-12-04
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Bulgarian Winter Competition 2003

3450. Fast Width

Problem code: FASTW

When you want to get quick to some place in the city you don't often look for the shortest distance to there. Sometimes what is important is how width the street is. We will say that a route in the city had width is the width of the smallest street you will have to pass through. Now you are give the city network of streets and intersections and the width of each street and you are asked to provide the width of the widest path between intersection with number 1 and intersection with number N.

Input

On the first line of the input you will find two integers N (2 < N < 10000) the number of intersections in the city and M (1 < M < 100000) the number of streets in the city. On each of the next M lines you will get information about one street in the form of 3 integers I, J, W (1 < W < 65000) which will mean that intersections I and J are connected via street with width W.

Output

A single integer representing the width of the path between 1 and N with maximum width. If no path exists between 1 and N output 0

Example

```
Input:
5 6
1 2 1
1 3 3
1 4 9
2 5 10
3 5 4
4 5 2

Output:
3
```

Added by:	Nikola P Borisov
Date:	2008-12-04
Time limit:	20s
Source limit:	50000B
Languages:	All
Resource:	Bulgarian National Olympiad Round 2 2004 Group A

SPOJ Problem Set (main)

3610. BOI 97 - Factorial

Problem code: BOIFAC

For a positive integer number N, find all positive integer numbers X (if any such number exists) with the property that the number 1*2*3*...*X has exactly N decimal digits. Assume that N is at most 150,000.

Input

A single line which contains a positive integer number denoting the number N.

Output

The first line should contain the string "NO", if such a number does not exist. Otherwise, the first line should contain a positive integer denoting how many X numbers exist. Then print all the X numbers, one number per line.

Example

```
Input:
5

Output:
1
8
```

Added by: Mir Wasi Ahmed
Date: 2008-12-31
Time limit: 1s
Source limit:50000B
Languages: C C99 strict C++ PAS gpc PAS fpc JAVA
Resource: Balkan Olympiad of Informatics 1997

4108. BOI 97 - Street Network

Problem code: BOI97SN

The street network of a city is composed of streets and nodes. In a node, two or more streets can meet. All streets are one-way streets. Note also that, two nodes can be connected directly by more than one street, and one node can have a street that loops back to itself. Write a computer program in order to address the following issues: 1. Is it possible to start from at least one node A and visit ALL streets exactly once ? 2. How many nodes can serve as starting points in order to satisfy the property of the previous case ? 3. For each node X, how many paths of length S exist starting from X and ending to X, where any street or node can be visited more than once ?

Input

In the first line in the input is a positive integer number N (N<=50), denoting the number of nodes in the city street network. The second line contains a positive integer number S (S<=3) denoting the path length. The next N lines contain the network description in matrix form. More precisely, the element in row I and column J is the number of streets from node I to node J.

Output

The first line contains the string "YES" if you can start from a node, travel through all streets exactly once, and arrive either at the starting point, or at another node. Otherwise, the string "NO" should appear in the output. If the answer is "YES", the next line of the output file should contain a positive integer number denoting how many nodes can serve as starting points. Finally, the last line of the output file should contain N positive integers (separated by a space) that show for each node how many different paths with length S exist such that each path leads from the node back to itself. These numbers should be sorted in increasing order.

Example

Input 1:
```
3
2
1 1 0
1 1 1
0 1 1
```

Output:
```
YES
3
2 2 3
```

Input 2:
```
3
2
1 1 0
1 1 2
0 0 1
```

Output:
```
NO
1 2 2
```

Added by:	Mir Wasi Ahmed
Date:	2009-03-24
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Balkan Olympiad of Informatics 1997

SPOJ Problem Set (partial)

5167. Quadratics

Problem code: BRHQUADR

Butch needs help with checking his math homework. He is studying quadratic equations, which are in the form

$$y = ax^2+bx+c$$

He wants to give you a, b, c, and x (1 <= a, b, c, x <= 10), and asks you to find y.

Input

Line 1: Three space-separated integers, a, b, and c.
Line 2: A single integer, x.

Output

Line 1: A single integer, y

Example

```
Input:2 5 3-4Output:
15
```

Added by:	Damon Doucet
Date:	2009-11-02
Time limit:	5s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (partial)

5201. Stacks of Bricks

Problem code: SBRICKS

Problem statement

You are given a sequence of *n* (*n* < 100) integers. Each number denotes the height of a stack of bricks. If we put the stacks in a line as in the illustration below, we would see stacks of uneven heights. Suppose a "move" is made by picking up one brick from one stack and putting it on another, compute the minimum number of moves to rearrange the bricks such that all stacks have the same height.

Read the input from standard input. The first line of the input is the integer *n*, followed by *n* lines of integers denoting the height of the *n* stacks. The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height. Your output to standard output should consist of exactly one integer denoting the minimum number of moves.

Sample input

```
6524175
```

Sample output

```
5
```

Added by:	Zhang Zhiyong Melvin
Date:	2009-11-04
Time limit:	1s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (partial)

5202. Stacks of Bricks 2

Problem code: SBRICKS2

Summary

This is similar to "Stacks of Bricks" except that for each move you are only allowed to move a brick to a stack on its immediate left or right.

Problem statement

You are given a sequence of *n* (*n* < 100) integers. Each number denotes the height of a stack of bricks. If we put the stacks in a line as in the illustration below, we would see stacks of uneven heights. Suppose a "move" is made by picking up one brick from one stack and putting it on *stack to its immediate left or right*, compute the minimum number of moves to rearrange the bricks such that all stacks have the same height.

Read the input from standard input. The first line of the input is the integer *n*, followed by *n* lines of integers denoting the height of the *n* stacks. The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height. Your output to standard output should consist of exactly one integer denoting the minimum number of moves.

Sample input

```
6
5
2
4
1
7
5
```

Sample output

```
8
```

Added by: Zhang Zhiyong Melvin
Date: 2009-11-04
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (partial)

6409. Suffix Array

Problem code: SARRAY

Given a string of length at most 100,000 consist of alphabets and numbers. Output the suffix array of the string.

A suffix array is an array of integers giving the starting positions (0-based) of suffixes of a string in lexicographical order. Consider a string "abracadabra0AbRa4Cad14abra". The size of the suffix array is equal to the length of the string. Below is the list of 26 suffixes of the string along with its starting position sorted in lexicographical order:

```
POS SUFFIX 11 0AbRa4Cad14abra
 20 14abra
 16 4Cad14abra
 21 4abra
 12 AbRa4Cad14abra
 17 Cad14abra
 14 Ra4Cad14abra
 25 a
 10 a0AbRa4Cad14abra
 15 a4Cad14abra
 22 abra
  7 abra0AbRa4Cad14abra
  0 abracadabra0AbRa4Cad14abra
  3 acadabra0AbRa4Cad14abra
 18 ad14abra
  5 adabra0AbRa4Cad14abra
 13 bRa4Cad14abra
 23 bra
  8 bra0AbRa4Cad14abra
  1 bracadabra0AbRa4Cad14abra
  4 cadabra0AbRa4Cad14abra
 19 d14abra
  6 dabra0AbRa4Cad14abra
 24 ra
  9 ra0AbRa4Cad14abra
  2 racadabra0AbRa4Cad14abra
```

Note: this is a partial score problem.
O(n^2 log(n)) is expected to score about 20-30. (Naive sorting all suffixes)
O(n $\log^2$(n)) is expected to score about 40. (OK for most programming contest problems)
O(n log n) is expected to score about 60-70. (Use counting sort for small alphabet size)
O(n) without tweaks is expected to score about 80-90.
O(n) with tweaks is expected to score 100. (This is meant for fun only :)

Input

A single line containing the string.

Output

The suffix array of the string.

Example

Input:
```
abracadabra0AbRa4Cad14abra
```

Output:
```
11
20
16
21
12
17
14
25
10
15
22
7
0
3
18
5
13
23
8
1
4
19
6
24
9
2
```

Added by: Felix Halim
Date: 2010-03-26
Time limit: 0.25s
Source limit:50000B
Languages: All except: PERL 6

SPOJ Problem Set (oi)

6610. Căt ba?ng

Problem code: CUTBOARD

SNAD muôn căt môt ba?ng hinh chu+~ nhât kích thu+o+'c M x N thanh các hinh chu+~ nhât con chie^'u dai hoăc chie^'u rông la 1. Môi lân căt nhu+ vây SNAD chi? có the^? căt tren ria cu?a ba?ng hinh chu+~ nhât đó, có nghia la có the^? cho.n 1 trong 4 phu+o+ng án: căt dong đâu tien, căt dong cuôi cung, căt côt đâu tien, căt côt cuôi cung. Tuy nhien tông các sô trong hinh chu+~ nhât con nhu+ vây không đu+o+.c vu+o+'t quá P. Ba.n hay giúp SNAD tim cách căt ba?ng ban đâu thanh ít hinh chu+~ nhât con nhât có the^?

Input

Dong đâu gôm 3 sô P, N, M (P < 200000000, 0 < M, N < 2001)
M dong sau mô ta? ma trân hinh chu+~ nhât kích thu+o+'c M x N, các sô tren ma trân nho? ho+n 100000

Output

Ghi sô duy nhât la ke^'t qua? nho? nhât có the^?

Example

`Input:``12 6 46 0 4 8 0 50 4 5 4 6 00 5 6 5 6 05 4 0 0 5 4`**`Output:`**`8`

Added by: Tran Hai Dang
Date: 2010-05-04
Time limit: 5s
Source limit:50000B
Languages: All except: TCL SCALA ERL TECS JS
Resource: POI 2005

SPOJ Problem Set (oi)

6652. Tru+o+.t tuye^'t phien ba?n 1

Problem code: SKIVER1

Vu+o+ng quôc no. có N thanh phô, bie^'t răng thanh phô 1 chi? có the^? đe^'n đu+o+.c các thanh phô khác ma không thanh phô nao đe^'n đu+o+.c thanh phô 1, thanh phô N không đe^'n đu+o+.c thanh phô nao khác ma chi? có thanh phô khác đe^'n thanh phô nay. Trong vu+o+ng quôc 2 thanh phô bât ki đu+o+.c nôi vo+'i nhau thông qua không quá 1 con đu+o+'ng môt chie^'u. Quanh năm vu+o+ng quôc ngâp trong tuye^'t, nha vua muôn tuye^?n môt đôi tru+o+.t tuye^'t gôm ít ngu+o+'i nhât sao cho môi ngu+o+'i đe^'u xuât phát tu+' thanh phô 1, qua các thanh phô trung gian rôi du+'ng la.i ta.i thanh phô N tho?a man các con đu+o+'ng môt chie^'u đu+o+.c thăm bo+?i ít nhât 1 thanh vien trong đôi.

Du+~ lie^.u

- Dong đâu ghi sô nguyen du+o+ng N, sô thanh phô (N < 1001).
- N -1 dong sau môi dong mô ta? thông tin ve^' môi thanh phô tu+' 1 đe^'n N - 1, sô đâu tien cu?a môi dong ghi sô K la lu+o+.ng thanh phô ma thanh phô đó có the^? đi to+'i, K sô tie^'p theo la chi? sô các thanh phô đó.

Ke^'t qu?a

Ghi ra sô nho? nhât các thanh vien trong đôi.

Ví du.

```
Du+~ lie^.u:
15
5 3 5 9 2 4
1 9
2 7 5
2 6 8
1 7
1 10
2 14 11
2 10 12
2 13 10
3 13 15 12
2 14 15
1 15
1 15
1 15
Ke^'t qu?a
8
```

Added by:	Tran Hai Dang
Date:	2010-05-15
Time limit:	1s
Source limit:	50000B
Languages:	All except: TCL SCALA ERL TECS JS
Resource:	Chút thay đôi tu+‘ bai POI 2001

SPOJ Problem Set (oi)

6671. Xe buýt

Problem code: MBUS

Tren măt phăng to.a đô cho N đie^?m la N tra.m xe buýt, o+? tra.m thu+' i có to.a đô (xi, yi) va có zi ngu+o+'i đang cho+' o+? đó. Cân đi xe buýt tu+' (1, 1) đe^'n (X, Y) sao cho đón đu+o+.c nhie^'u ngu+o+'i nhât có the^?, bie^'t răng xe buýt chi? có the^? đi len tren hoăc sang pha?i.

Du+~ lie^.u

- Dong đâu ghi 3 sô nguyen du+o+ng X, Y, N (X, Y < 1000000001, N < 100001)
- N dong sau môi dong ghi 3 sô xi, yi, zi (xi <= X, yi <= Y, zi < 1000001).

Ke^'t qu?a

Ghi tren môt dong sô duy nhât la sô khách lo+'n nhât có the^? đón đu+o+.c.

Ví du.

```
Du+~ lie^.u:
8 7 11
4 3 4
6 2 4
2 3 2
5 6 1
2 5 2
1 5 5
2 1 1
3 1 1
7 7 1
7 4 2
8 6 2Ke^'t qu?a
11
```

Added by:	Tran Hai Dang
Date:	2010-05-21
Time limit:	1s
Source limit:	50000B
Languages:	All except: TCL SCALA ERL TECS JS
Resource:	POI 2005

SPOJ Problem Set (classical)

8094. Treasure map

Problem code: SPMAP

Yk is a archaeologist. When discovering the pyramids of Egypt, he found a treasure map which show the location of n secret islands, each island has only a kind of precious stone, with the weight m[i](kg) and the quantity s[i].

So he decided to buy a plane to look for the treasure. But he just hired a small plane which carrying the maximum of l(kg).

So he want to know how many ways to select the stones which fill the plane? (total of the weight of the stones is l).

Input

- The first line is two integer: l,n (l<=20000 , n<=500).

- Each of next n line is two integer: m[i],s[i] (m[i]<=5000 , s[i]<=100).

Output

- Only line is your answer.

Example

```
Input:10 34 74 52 5Output:6Note: you must optimize your code to get AC, time limit will be 11s :)
```

Added by:	?[HEART]?- yk -?[HEART]?
Date:	2010-12-18
Time limit:	11s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (partial)

8760. Salesman

Problem code: SALESMAN

Môt thu+o+ng nhân vu+‘a quye^’t đi.nh mua môt con tau mo+’i phu.c vu. cho các cuôc trao đôi hang hóa do.c bo+‘ sông Danube. Con tau cu?a anh ta có tôc đô rât tuye^.t vo+‘i, nó có the^? đu+a anh ta đe^’n bât ki vi. trí nao tren sông trong không quá môt tích tăc, nhu+ng, đó la.i la môt con tau rât tôn nhien lie^.u. Con tau tôn U dollar cho 1 mét đi ngu+o+.c dong, va mât D dollar cho 1 mét đi xuôi dong.
Có tât ca? N hôi cho+. săp su+?a die^~n ra tren khăp bo+‘ sông Danube. Môi hôi cho+. chi? die^~n ra trong 1 ngay. Vo+’i môi hôi cho+. x, thu+o+ng nhân bie^’t hôi cho+. băt đâu vao ngay T[x] (tính tu+‘ khi mua tau), ta.i vi. trí cách thu+o+.ng nguôn L[x] mét, va ne^’u tham gia, thu+o+ng nhân se~ kie^’m đu+o+.c M[x] dollar. Anh ta se~ pha?i băt đâu va ke^’t thúc chuye^’n đi cu?a minh ta.i môt vi. trí cách thu+o+.ng nguôn S mét.
Lu+u ý răng, các hôi cho+. băt đâu theo thu+’ tu+. tho+‘i gian, nen ne^’u nhu+ hôi cho+. A đu+o+.c băt đâu so+’m ho+n hôi cho+. B, thu+o+ng nhân không the^? tham gia hôi cho+. B tru+o+’c hôi cho+. A. Tuy nhien, ne^’u nhu+ có nhie^‘u hôi cho+. die^~n ra trong cung 1 ngay, thu+o+ng nhân có the^? tham gia bât ki hôi cho+. nao theo bât ki thu+’ tu+. nao anh ta muôn. Thu+o+ng nhân có the^? đi qua 1 hôi cho+. nhie^‘u lân trong ngay, nhu+ng di nhien, anh ta se~ chi? kie^’m đu+o+.c tie^‘n trong lân đâu tien ghé qua hôi cho+. đó.
Nhie^.m vu. cu?a ba.n, la hay vie^’t chu+o+ng trinh giúp thu+o+ng nhân tính đu+o+.c sô tie^‘n nhie^‘u nhât thu đu+o+.c, khi tham gia các hôi cho+. theo phu+o+ng án tôi u+u. Sô tie^‘n kie^’m đu+o+.c, băng tông lo+.i nhuân đa.t đu+o+.c tu+‘ nhu+~ng hôi cho+. anh ta tham gia, tru+‘ đi chi phí di chuye^?n tren sông.
Input:
- Dong đâu tien gôm 4 sô nguyen N,U,D,S
- N dong tie^’p theo, môi gôm 3 sô T[k], L[k], M[k] mô ta? hôi cho+. k: hôi cho+. băt đâu vao ngay T[k], cách thu+o+.ng nguôn L[k] mét, va lo+.i nhuân đa.t đu+o+.c ne^’u tham gia la M[k]
- Các sô tren cung 1 dong cách nhau bo+?i ít nhât 1 dâu cách
Output
- Gôm 1 sô nguyen duy nhât, la sô tie^‘n lo+’n nhât ma thu+o+ng nhân có the^? kie^’m đu+o+.c (giá tri. nay có the^? băng 0).
Gio+’i ha.n:
- 1 <= N <= 500000, sô hôi cho+.
- 1 <= D <= U <= 10, chi phí di chuye^?n 1 mét ngu+o+.c dong (U), va xuôi dong (D)
- 1 <= S <= 500001, vi. trí xuât phát cu?a thu+o+ng nhân
- 1 <= T[k] <= 500000, ngay die^~n ra hôi cho+. thu+’ k
- 1 <= L[k] <= 500001, vi. trí cu?a hôi cho+. thu+’ k
- 1 <= M[k] <= 4000, sô tie^‘n thu+o+ng nhân có the^? kie^’m đu+o+.c ne^’u tham gia hôi cho+. thu+’ k
- Không có hai hôi cho+. nao đu+o+.c mo+? o+? cung vi. trí, cung nhu+ không có hôi cho+. nao đu+o+.c mo+? ta.i nha cu?a thu+o+ng nhân
Example:
Input
4 5 3 100
2 80 100

20 125 130
10 75 150
5 120 110
Output
50
Gia?i thích:
Phu+o+ng án tôi u+u cu?a thu+o+ng nhân se~ la tham gia hôi cho+. 1 va 3, theo thu+' tu+. nhu+ sau
- Đi ngu+o+.c dong 20 mét, chi phí 100 dollar. Lo+.i nhuân hie^.n ta.i: -100
- Tham gia hôi cho+. 1, kie^'m đu+o+.c 100 dollar. Lo+.i nhuân hie^.n ta.i: 0
- Đi ngu+o+.c dong 5 mét, chi phí 25 dollar. Lo+.i nhuân hie^.n ta.i: -25
- Tham gia hôi cho+. 3, kie^'m đu+o+.c 150 dollar. Lo+.i nhuân hie^.n ta.i: 125
- Đi xuôi dong 25 mét va tro+? ve^` nha, chi phí 75 dollar. Lo+.i nhuân cuôi cung: 50 dollar

---

Added by:	sieunhan
Date:	2011-04-21
Time limit:	12s
Source limit:	50000B
Languages:	All
Resource:	IOI 2009, day 2