Problem 1: Marathon

Unhappy with the poor health of his cows, Farmer John enrolls them in an assortment of different physical fitness activities. His prize cow Bessie is enrolled in a running class, where she is eventually expected to run a marathon through the downtown area of the city near Farmer John's farm!

The marathon course consists of N checkpoints (3 <= N <= 100,000) to be visited in sequence, where checkpoint 1 is the starting location and checkpoint N is the finish. Bessie is supposed to visit all of these checkpoints one by one, but being the lazy cow she is, she decides that she will skip up to one checkpoint in order to shorten her total journey. She cannot skip checkpoints 1 or N, however, since that would be too noticeable.

Please help Bessie find the minimum distance that she has to run if she can skip up to one checkpoint.

Note that since the course is set in a downtown area with a grid of streets, the distance between two checkpoints at locations (x1, y1) and (x2, y2) is given by |x1-x2| + |y1-y2|. This way of measuring distance -- by the difference in x plus the difference in y -- is sometimes known as "Manhattan" distance because it reflects the fact that in a downtown grid, you can travel parallel to the x or y axes, but you cannot travel along a direct line "as the crow flies".

INPUT:

The first line gives the value of N.

The next N lines each contain two space-separated integers, x and y, representing a checkpoint (-1000 <= x <= 1000, -1000 <= y <= 1000). The checkpoints are given in the order that they must be visited. Note that the course might cross over itself several times, with several checkpoints occurring at the same physical location. When Bessie skips such a checkpoint, she only skips one instance of the checkpoint -- she does not skip every checkpoint occurring at the same location.

SAMPLE INPUT:

```
4
0 0
8 3
11 -1
10 0
```

OUTPUT:

Output the minimum distance that Bessie can run by skipping up to one checkpoint. Don't forget to end your output with a newline. In the sample case shown here, skipping the checkpoint at (8, 3) leads to the minimum total distance of 14.

SAMPLE OUTPUT:

```
14
```

Problem 2: Crosswords

Like all cows, Bessie the cow likes to solve crossword puzzles. Unfortunately, her sister Elsie has spilled milk all over her book of crosswords, smearing the text and making it difficult for her to see where each clue begins. It's your job to help Bessie out and recover the clue numbering!

An unlabeled crossword is given to you as an N by M grid (3 <= N <= 50, 3 <= M <= 50). Some cells will be clear (typically colored white) and some cells will be blocked (typically colored black). Given this layout, clue numbering is a simple process which follows two logical steps:

Step 1: We determine if a each cell begins a horizontal or vertical clue. If a cell begins a horizontal clue, it must be clear, its neighboring cell to the left must be blocked or outside the crossword grid, and the two cells on its right must be clear (that is, a horizontal clue can only represent a word of 3 or more characters). The rules for a cell beginning a vertical clue are analogous: the cell above must be blocked or outside the grid, and the two cells below must be clear.

Step 2: We assign a number to each cell that begins a clue. Cells are assigned numbers sequentially starting with 1 in the same order that you would read a book; cells in the top row are assigned numbers from left to right, then the second row, etc. Only cells beginning a clue are assigned numbers.

For example, consider the grid, where '.' indicates a clear cell and

'#' a blocked cell.

```
...
#..
...
..#
.##
```

Cells that can begin a horizontal or vertical clue are marked with ! below:

```
!!!
#..
!..
..#
.##
```

If we assign numbers to these cells, we get the following;

```
123
#..
4..
..#
.##
```

Note that crossword described in the input data may not satisfy constraints typically seen in published crosswords. For example, some clear cells may not be part of any clue.

INPUT: (file crosswords.in)

The first line of input contains N and M separated by a space.

The next N lines of input each describe a row of the grid. Each contains M characters, which are either '.' (a clear cell) or '#' (a blocked cell).

SAMPLE INPUT:

```
5 3
...
#..
...
..#
.##
```

OUTPUT: (file crosswords.out)

On the first line of output, print the number of clues.

On the each remaining line, print the row and column giving the position of a single clue (ordered as described above). The top left cell has position (1, 1). The bottom right cell has position (N, M).

SAMPLE OUTPUT:

```
4
1 1
1 2
1 3
```

3 1

Problem 3: Cow Jog

The cows are out exercising their hooves again! There are N cows jogging on an infinitely-long single-lane track (1 <= N <= 100,000). Each cow starts at a distinct position on the track, and some cows jog at different speeds.

With only one lane in the track, cows cannot pass each other. When a faster cow catches up to another cow, she has to slow down to avoid running into the other cow, becoming part of the same running group.

Eventually, no more cows will run into each other. Farmer John wonders how many groups will be left when this happens. Please help him compute this number.

INPUT: (file cowjog.in)

The first line of input contains the integer N.

The following N lines each contain the initial position and speed of a single cow. Position is a nonnegative integer and speed is a positive integer; both numbers are at most 1 billion. All cows start at distinct positions, and these will be given in increasing order in the input.

SAMPLE INPUT:

```
5
0 1
1 2
```

2 3

3 2

6 1

OUTPUT: (file cowjog.out)

A single integer indicating how many groups remain.

SAMPLE OUTPUT:

2

Problem 4: Learning by Example

Farmer John has been reading all about the exciting field of machine learning, where one can learn interesting and sometimes unexpected patterns by analyzing large data (he has even started calling one of the fields on his farm the "field of machine learning"!). FJ decides to use data about his existing cow herd to build an automatic classifier that can guess whether a cow will have spots or not.

Unfortunately, FJ hasn't been very good at keeping track of data about his cows. For each of his N cows (1 <= N <= 50,000), all he knows is the weight of the cow, and whether the cow has spots. Each of his cows has a distinct weight. Given this data, he builds what is called a "nearest neighbor classifier". To guess whether a new cow C will have spots or not, FJ first finds the cow C' in his herd with weight closest to that of C. If C' has spots, then FJ guesses that C will also have spots; if C' has no spots, FJ guesses the same for C. If there is not one unique nearest neighbor C' but rather a tie between two of FJ's cows, then FJ guesses that C will have spots if one or both these nearest neighbors has spots.

FJ wants to test his new automatic spot predictor on a group of new cows that are just arriving at his farm. After weighing these cows, he sees that the new shipment of cows contains a cow of every integer weight between A and B (inclusive). Please determine how many of these cows will be classified as having spots, using FJ's new classifier. Note that the classifier only makes decisions using data from FJ's N existing cows, not any of the new cows. Also note that since A and B can both be quite large, your program will not likely run fast enough if it loops from A to B counting by ones.

INPUT: (file learning.in)

The first line of the input contains three integers N, A, and B (1 <= A <= B <= 1,000,000,000).

The next N lines each describe a single cow. Each line contains either S W, indicating a spotted cow of weight W, or NS W, indicating a non-spotted cow of weight W. Weights are all integers in the range 1 ... 1,000,000,000.

SAMPLE INPUT:

```
3 1 10
S 10
NS 4
S 1
```

OUTPUT: (file learning.out)

A single integer giving the number of incoming cows that FJ's algorithm will classify as having spots. In the example shown here, the incoming cows of weights 1, 2, 7, 8, 9, and 10 will all be classified as having spots.

SAMPLE OUTPUT:

```
6
```

Problem 5: Piggy Back

Bessie and her sister Elsie graze in different fields during the day, and in the evening they both want to walk back to the barn to rest. Being clever bovines, they come up with a plan to minimize the total amount of energy they both spend while walking.

Bessie spends B units of energy when walking from a field to an adjacent field, and Elsie spends E units of energy when she walks to an adjacent field. However, if Bessie and Elsie are together in the same field, Bessie can carry Elsie on her shoulders and both can move to an adjacent field while spending only P units of energy (where P might be considerably less than B+E, the amount Bessie and Elsie would have spent individually walking to the adjacent field). If P is very small, the most energy-efficient solution may involve Bessie and Elsie traveling to a common meeting field, then traveling together piggyback for the rest of the journey to the barn. Of course, if P is large, it may still make the most sense for Bessie and Elsie to travel separately. On a side note, Bessie and Elsie are both unhappy with the term "piggyback", as they don't see why the pigs on the farm should deserve all the credit for this remarkable form of transportation.

Given B, E, and P, as well as the layout of the farm, please compute the minimum amount of energy required for Bessie and Elsie to reach the barn.

INPUT:
The first line of input contains the positive integers B, E, P, N, and M. All of these are at most 40,000. B, E, and P are described above.

N is the number of fields in the farm (numbered 1..N, where N >= 3), and M is the number of connections between fields. Bessie and Elsie start in fields 1 and 2, respectively. The barn resides in field N.

The next M lines in the input each describe a connection between a pair of different fields, specified by the integer indices of the two fields. Connections are bi-directional. It is always possible to travel from field 1 to field N, and field 2 to field N, along a series of such connections.

SAMPLE INPUT:

```
4 4 5 8 8
1 4
2 3
3 4
4 7
2 5
5 6
6 8
7 8
```

OUTPUT:

A single integer specifying the minimum amount of energy Bessie and Elsie collectively need to spend to reach the barn. In the example shown here, Bessie travels from 1 to 4 and Elsie travels from 2 to 3 to 4. Then, they travel together from 4 to 7 to 8.

SAMPLE OUTPUT:

```
22
```

Problem 6: Guard Mark

Farmer John and his herd are playing frisbee. Bessie throws the frisbee down the field, but it's going straight to Mark the field hand on the other team! Mark has height H (1 <= H <= 1,000,000,000), but there are N cows on Bessie's team gathered around Mark (2 <= N <= 20). They can only catch the frisbee if they can stack up to be at least as high as Mark. Each of the N cows has a height, weight, and strength. A cow's strength indicates the maximum amount of total weight of the cows that can be stacked above her.

Given these constraints, Bessie wants to know if it is possible for her team to build a tall enough stack to catch the frisbee, and if so, what is the maximum safety factor of such a stack. The safety factor of a stack is the amount of weight that can be added to the top of the stack without exceeding any cow's strength.

INPUT: (file guard.in)

The first line of input contains N and H.

The next N lines of input each describe a cow, giving its height, weight, and strength. All are positive integers at most 1 billion.

SAMPLE INPUT:

4 10
9 4 1
3 3 5
5 5 10

4 4 5

OUTPUT: (file guard.out)

If Bessie's team can build a stack tall enough to catch the frisbee, please output the maximum achievable safety factor for such a stack. Otherwise output "Mark is too tall" (without the quotes).

SAMPLE OUTPUT:

2

PROBLEM 7. CENSORING

Farmer John has purchased a subscription to Good Hooveskeeping magazine for his cows, so they have plenty of material to read while waiting around in the barn during milking sessions. Unfortunately, the latest issue contains a rather inappropriate article on how to cook the perfect steak, which FJ would rather his cows not see (clearly, the magazine is in need of better editorial oversight).

FJ has taken all of the text from the magazine to create the string S of length at most 10^5 characters. He has a list of censored words t_1 ... t_N that he wishes to delete from S. To do so Farmer John finds the earliest occurrence of a censored word in S (having the earliest start index) and removes that instance of the word from S. He then repeats the process again, deleting the earliest occurrence of a censored word from S, repeating until there are no more occurrences of censored words in S. Note that the deletion of one censored word might create a new occurrence of a censored word that didn't exist before.

Farmer John notes that the censored words have the property that no censored word appears as a substring of another censored word. In particular this means the censored word with earliest index in S is uniquely defined.

Please help FJ determine the final contents of S after censoring is complete.

INPUT FORMAT:

The first line will contain S. The second line will contain N, the number of censored words. The next N lines contain the strings t_1 ... t_N. Each string will contain lower-case alphabet characters (in the range a..z), and the combined lengths of all these strings will be at most 10^5.

OUTPUT FORMAT:

The string S after all deletions are complete. It is guaranteed that S will not become empty during the deletion process.

SAMPLE INPUT:

```
begintheescapexecutionatthebreakofdawn
2
escape
execution
```

SAMPLE OUTPUT:

```
beginthatthebreakofdawn
```