2016 Mid-Atlantic Regional Programming Contest

Welcome to the 2016 ICPC Mid-Atlantic Regional. Before you start the contest, please take the time to review the following:

The Contest

1. There are eight (8) problems in the packet, labeled A–H. These problems are NOT sorted by difficulty. As a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Periodic Strings	Orange
B	Around and Around We Go	Green
C	Buggy Robot	Silver
D	Enclosure Area	Pink
E	Toys and Triangles	Red
F	Islands	Yellow
G	Ghostbusters 2	Black
H	Painting the Floodwall	Purple

2. The winning team is the one that successfully completes the most problems in the time allowed.

   If teams are tied with the same number of problems solved, the tie is broken in favor of the team with the fewest penalty points. For each problem *solved correctly*, penalty points are charged as the sum of

   - the number of minutes elapsed since the start of the contest to when the successful submission was made, and
   - 20 points for each incorrect submission prior to the successful one.

   No penalty points are added for problems that are never solved.

3. In the event that you feel a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification.

   If a clarification is issued during the contest, it will be broadcast to *all* teams.

   If the judges believe that the problem statement is sufficiently clear, you will receive the response, "No response, read problem statement." If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you may have found.

Submitting

4. Solutions for problems submitted for judging are called runs. Each run will be judged.

5. The judges will execute your program on multiple test cases. To be successful, each test execution must return the correct output, formatted as specified in the problem statement, and must complete execution within the appropriate time limit.

   If a problem does not specify a time limit as part of its output format description, then it must complete each test case within 2 seconds.

   - Although Python is accepted as a programming language in this contest, no guarantee is made that a Python solution is possible that runs within the time limits allowed for any given problem.

6. The judges will respond to your submission with one of the following responses.

Response	Explanation
Yes	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Incomplete Output	Your submission did not produce all of the required output.
Excessive Output	Your submission generated output in addition to or instead of what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve one or more of the judges' test data within the allotted time period.
Other-Contact Staff	Contact your local site judge for clarification.

7. In the event that more than one response is applicable, the judges may respond with any of the applicable responses. For example, a program that runs too long but produces incorrect output before it is killed might receive either a "Wrong Answer" or a "Time-Limit Exceeded" response. A program that crashes before completing the test data set might receive either an "Incomplete Output" or a "Run-Time Error" response.

8. **Do not** request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, **you may have a runner ask the local site judge to determine the cause of the delay.**

   If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

9. The submission of code deliberately designed to delay, crash, or otherwise negatively affect the contest itself will be considered grounds for immediate disqualification.

Your Programs

10. Your program must be contained within a single source-code file. Java programs should be written in the default (unnamed) package, meaning that it should not contain a `package` statement at all.

    Use the filename extension "`.cpp`" for C++ program files. Use the extension "`.c`" for C program files. Use the extension "`.java`" for Java program files. Use the extension "`.py`" for Python program files.

    Note that all filename extensions are lower case.

11. Your code will be compiled for judging as follows:

    C: `gcc -O2 -std=gnu99 -static` *yourFileName* `-lm`

    C++: `g++ -O2 -std=gnu++11` *yourFileName*

    Java: `javac -encoding UTF-8 -sourcepath .  -d .` *yourFileName*

    For Java, the compiled code will be executed using the command:

    `java XX:+UseSerialGC -Xss64m -Xms1024m -Xmx1024m` *yourClassName*

    Python 3: `python3` *yourFileName*

12. All solutions must read from standard input and write to standard output. Do not print to the standard error stream, as this may be mistaken for a run-time error signal.

13. Unless otherwise specified, all lines of program output

    - must be left justified, with no leading blank spaces prior to the first non-blank character on that line,
    - must end with the appropriate line terminator (`\n`, `endl`, or `println()`), and
    - must not contain any blank characters at the end of the lines, between the final specified output and the line terminator.

    You must not print extra lines of output, even if empty, that are not specifically required by the problem statement.

14. Unless otherwise specified, all numbers in your output should begin with the minus sign (-) if negative, followed immediately by 1 or more decimal digits. If the number being printed is a floating point number, then the decimal point should appear, followed by the appropriate number of decimal digits.

    For output of real numbers, the number of digits after the decimal point will be specified in the problem description (as the "*decimal digits of precision*").

    All floating point numbers printed to a given precision should be rounded to the nearest value, using unbiased (a.k.a. half-to-even) rounding. For example, if 2 decimal digits of precision is requested, then the table below shows how certain exact values would be printed:

exact	prints as
0.0149	0.01
0.0251	0.03
0.015	0.02
0.025	0.02

In other words, neither scientific notation nor commas will be used for numbers, and you should ensure that you use a printing technique that rounds to the appropriate precision.

15. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

16. All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

17. If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points and the following digits may be omitted for numbers with no non-zero decimal portion. Scientific notation will not be used in input sets unless a problem explicitly allows it.

18. Every effort has been made to ensure that the compilers and run-time environments used by the judges are as similar as possible to those that you will use in developing your code. With that said, some differences may exist. It is, in general, your responsibility to write your code in a portable manner compliant with the rules and standards of the programming language. You should not rely upon undocumented and non-standard behaviors.

Good luck, and HAVE FUN!!!

Problem A: Periodic Strings

Define a *k-periodic string* as follows:

> A string s is k-periodic if $|s|$, the length of the string, is a multiple of k and, if you chop the string up into $|s|/k$ substrings of length k, then each of those substrings (except the first) is the same as the previous substring, but with its last character moved to the front.

For example, the following string is 3-periodic:

$$abccabbcaabc$$

The above string breaks up into substrings abc, cab, bca, and abc, and each substring (except the first) is a rotation of the previous substring ($abc \rightarrow cab$, $cab \rightarrow bca$, $bca \rightarrow abc$).

Given a string, determine the smallest k for which the string is k-periodic.

Input

Input will be a single line containing a string s, ($1 \leq |s| \leq 100$), consisting only of lower-case letters.

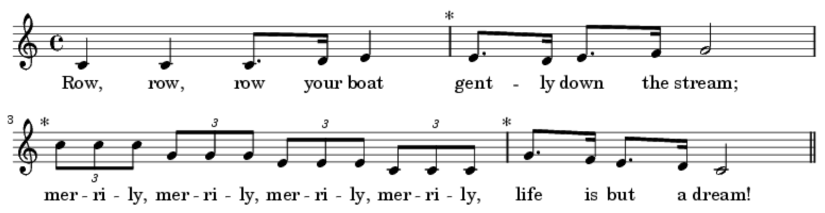
Output

Print a single line containing an integer denoting the smallest value of k for which the input string is k-periodic.

Examples

Sample Input	Sample Output
aaaaaaaa	1
abbaabbaabba	2
abcdef	6
abccabbcaabc	3

Problem B: Around and Around We Go


Row, row, row your boat gent - ly down the stream;
mer - ri - ly, mer - ri - ly, mer - ri - ly, mer - ri - ly, life is but a dream!

A *round* is a musical arrangement in which two or more voices repeat the same melodic line at times offset from one another. One of the more famous rounds is the nursery rhyme, "Row, Row, Row Your Boat", shown here, where a new voice can enter when the first voice reaches any of the '*'s.

In western music, an integer count of time units can be assigned to each syllable of a word to indicate how long that syllable is sung. Occasionally a pause (called a "rest") of one or more time units is inserted, and is also assigned an integer count of time units. If you know the time allocated to each word and rest, and the time offset at which each next voice begins singing the line, you can figure out which words will overlap in the round.

Write a program to list a two-voice round, lining up the words in the two voices so that syllables being sung simultaneously appear in a vertical column.

For each line of the song, print that line for the first voice, followed by a line for the second voice. The first voice line will contain the same syllables as the original line of input. Implicitly, that line defines a period of time starting with the beginning of the sound or the first syllable and ending of the sound of the last syllable on that line. The second voice line will contain any syllables of the song that start their sound during that time period (though the last syllable might not finish sounding until afterwards).

Each syllable will be printed as far to the left as possible, with plus signs ('+') inserted in front of syllables so as to satisfy the constraints:

- Consecutive symbols in the same line will be separated by at least one '+'.
- The first syllable in a line for the first voice is printed starting in the leftmost column of the line.
- Two syllables that begin at the same time in their respective voices are printed with their leftmost characters in the same column.
- If a syllable S2 in the either voice begins K time units after a syllable S1 in the other voice begins, then the first character of S2 appears in a column at least K to the right of the first letter of S1.

It is possible that some second-voice lines will be empty (if no syllables of the second voice are started during that line).

It is possible (in fact, likely) that not all syllables of the second voice will be printed.

Input

Input will begin with a line containing two integers, L and N. L is the number of lines in the song, $0 < L \leq 10$. N indicates the time at which the second voice begins, assuming that the first voice begins at time zero. $0 \leq N \leq 128$.

That will be followed by L pairs of lines.

The first line in each pair contains the syllables of that line of the song. A syllable is a string of any non-whitespace characters. Adjacent syllables in the input will be separated by a single blank. These lines will be at most 80 characters long.

The second line in each pair will consist of positive integers, one per syllable from the first line of the pair, indicating the time allocated to the corresponding syllables. These will be in the range $1 \ldots 128$.

Output

For each dataset, print $2L$ lines of output corresponding to two voices in a round, as described above.

Example

For the input:

```
2 8
Hot cross buns! = Hot cross buns! =
2 2 2 2 2 2 2 2
One a pen- ny, Two a pen- ny, Hot cross buns! =
1 1 1 1 1 1 1 1 2 2 2 2
```

the output should be

```
Hot+cross+buns!+=+Hot+cross+buns!+=
++++++++++++++++++Hot+cross+buns!+=
One+a+pen-+ny,+Two+a+pen-+ny,+Hot+++cross++++buns!+=
Hot+++cross++++buns!+=++++++++One+a+pen-+ny,+Two+a+pen-+ny,
```

For the input:

```
5 32
And ev- ry-
1 1 1
one neath a vine and fig tree, = shall live in
2 1 1 2 2 2 2 1 1 1 1
peace and un- a- fraid. =
2 2 2 2 6 2
And in- to plow shares beat their swords.
```

```
2 1 1 2 2 2 2 4
Na- tions shall learn war no more =
2 2 2 2 2 2 1 3
```

the output should be

```
And+ev-+ry-

one+neath+a+vine+and+fig+tree,+=+shall+live+in

peace+and+un-+a-+fraid.+++=
++++++++++++++++++++++And+ev-+ry-
And+in-+++to+plow+shares+beat+their+swords.
one+neath+a++vine+and++++fig++tree,+=+shall+live+in
Na-+++tions+shall+learn+war+no+more+=
peace+and+++un-+++a-++++fraid.+++++++=
```

Problem C: Buggy Robot

There is a robot in a 2D grid. The grid consists of obstacles, and there is exactly one cell that is the exit. The robot will exit the grid if it ever reaches the exit cell. Empty cells are denoted as '`.`', the robot's initial position is denoted as '`R`', obstacles are denoted as '`#`', and the exit is denoted as '`E`'.

You can program the robot by sending it a series of commands. The series of commands is a string consisting of characters '`L`' (move one square left), '`U`' (move one square up), '`R`' (move one square right), or '`D`' (move one square down). If the robot would run into an obstacle or off the edge of the grid, it will ignore the command (but it will continue onto future commands, if there are any).

Your friend sent a series of commands to the robot, but unfortunately, the commands do not necessarily take the robot to the exit.

You would like to fix the string so that the robot will touch the exit square. (Note: once the robot reaches the exit, it stops, even if there are more commands in the string.)

You can fix the string with a sequence of operations. There are two operations: inserting a command or deleting a command. What is the minimum number of operations you would need to fix the program?

Input

The first line of the input contains the two integers N and M, $1 \leq N, M \leq 50$, which are the width and height of the grid.

The next N lines will contain a string of exactly M characters, each of which is '.' (empty), '`R`' (the robot), '`#`' (an obstacle), or '`E`', the exit. There will be exactly one '`R`' and one '`E`' in the grid, and it will always be possible to navigate the robot to the exit.

The last line contains a string s ($1 \leq |s| \leq 50$) of commands. The string s will consist only of '`L`', '`R`', '`U`', and '`D`'.

Output

Print one line containing the single integer indicating the minimum number of operations necessary to fix the command string so that the robot makes it to the exit.

Example

Sample Input	Sample Output

`3 3`
`R..`
`.#.`
`..E`
`LRDD`
	1

`2 4`
`R.#.`
`#..E`
`RRUUDDRRUUUU`
	0

In the first example, we can insert the character R into the middle to get “LRRDD”.

In the second example, the robot will touch the exit cell during the given path, so we don’t need to do anything.

Problem D: Enclosure Area

No one really knew how rich old Mr. Miserly really was. People used to joke about how he must own half the town, but it was not until he passed away that people discovered that he was the sole owner of the town's banks. Since most of the homes and small businesses had mortgages with those banks, he really did own half the town.

No one really knew how civic-minded Mr. Miserly really was. But the day after his will was read, the townsfolk woke to the sound of workmen hammering wooden stakes into the ground near every building that was partly owned by the banks. Each family found, in their mailbox, a small green flag and a leaflet with these instructions:

> *From:* the executors of the Miserly estate
>
> You may tie this flag to any of the wooden stakes in the town. At noon, the executors will tie strings from one flagged stake to the next, forming the shortest path they can make that encloses *all* the flagged stakes. [The high school's Geometry teacher tried to explain to people that this formed a convex polygon, but no one paid much attention.]
>
> Any building inside the perimeter will have its mortgage paid off by the Miserly estate.
>
> Furthermore, the estate will donate $1 to the Municipal Fund for every square ILMU[1] of area within the enclosed perimeter.

Most of the townsfolk never got as far as reading the final paragraph before rushing off to place their flag near their home or family business.

You, however, held on to your flag. You now have the last flag. You can see that your own home is safe, surrounded by the flags of your neighbors. So your thoughts turn to the donation to the Municipal Fund. You want to place your flag where it will increase the enclosed area to the largest possible value.

Input

The first line of input contains two integers n and k, where n is the number of stakes in the forest and k is the number of stakes with flags already tied to them. $3 \leq k < n \leq 100{,}000$

The next n lines each have two integers x and y, $-1{,}000{,}000{,}000 \leq x, y \leq 1{,}000{,}000{,}000$, specifying the location of each stake, measured in ILMUs. The first k of these are the stakes already flagged by the townspeople.

- Assume that any two stakes can be joined by a straight length of string, ignoring any possible interference from buildings, trees, or other physical objects.
- There may be unflagged stakes both within and without the current perimeter.
- No three stakes will occupy co-linear locations.

[1]Imaginary Linear Measurement Unit

Output

Print a single line containing a floating point number denoting the maximum area you can achieve by gaining flagging one additional stake. Print this to one decimal place of precision.

Time Limit

Solutions to this problem must run in no more than 6 seconds.

Example

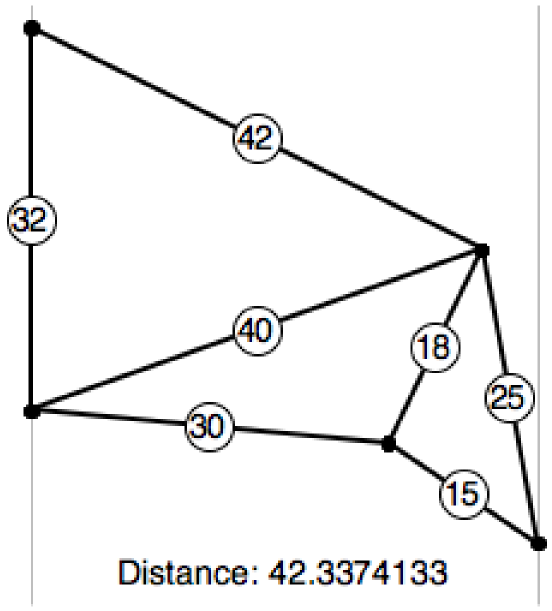
Sample Input	Sample Output

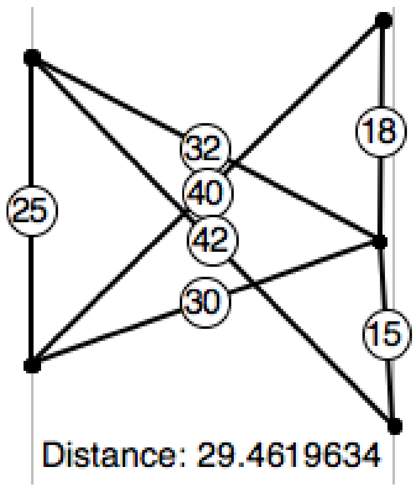
```
5 3
-5 -5
-5 5
5 -5
-4 6
5 5
```
	100.0

```
10 9
4 3
3 8
7 9
8 1
9 9
1 0
9 2
10 4
5 3
7 1
```
	58.5

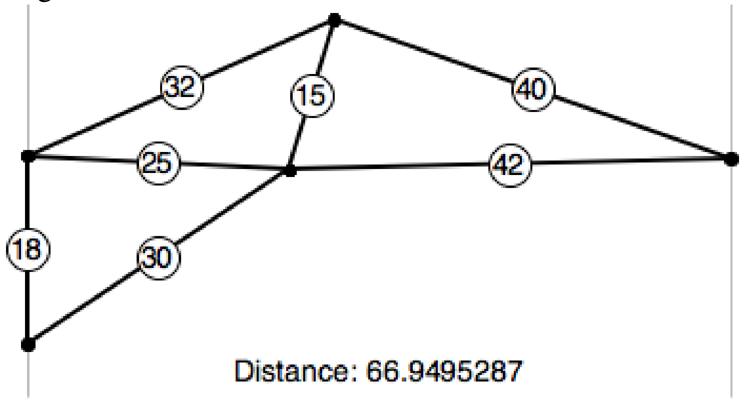
Problem E: Toys and Triangles

Alaa fondly remembers playing with a building toy when she was a child. It consisted of segments that could be fastened at each end. A game she liked to play was to start with one segment as a base, placed flat against a straight wall. Then she repeatedly added on triangles, with one edge of the next triangle being a single segment already in place on her structure, and the other two sides of the triangle being newly added segments. Of course no segment could go through the wall, but she did allow newly added segments to cross over already placed ones. Her aim was to see how far out from the wall she could make her structure go.





She would experiment, building different ways with different combinations of some or all of her pieces. It was an easy, boring task if all the segments that she used were the same length! It got more interesting if she went to the opposite extreme and started from a group of segments that were all of distinct lengths.



For instance, with segments of length 42, 40, 32, 30, 25, 18 and 15, the figures above show some of the structures she could have built, including the one with the maximum distance.

Now looking back as a Computer Science student, Alaa wondered how well she did, and decided to write a program to check what the maximum distance really is. This is also the challenge for you.

Input

The input is a single line of positive integers,

$$n \; L_1 \; L_2 \ldots L_n$$

Here n is the number of segments, $3 \leq n \leq 10$, and then the lengths of the segments are listed in strictly decreasing order. Each segment length is less than 100. At least one triangle may be constructed.

Output

Print the maximum distance that one of Alaa's structures can go from the wall to two decimal digits of precision.

Time Limit

Solutions to this problem must run in no more than 5 seconds.

Examples

Sample Input	Sample Output
3 50 40 30	40.00
4 50 40 30 29	40.00
7 42 40 32 30 25 18 15	66.95

Problem F: Islands

You are mapping a faraway planet using a satellite.

The planet's surface can be modeled as a grid. Your satellite has captured an image of the surface. Each grid square is either land (denoted as '`L`'), water (denoted as '`W`'), or covered by clouds ('`C`'). Clouds mean that the surface could either be land or water, but you can't tell.

An island is a maximal region of land where every grid cell in the island is reachable from every other by a path that only goes up, down, left or right.

Given an image, determine the minimum number of islands that is consistent with the given information.

Input

The first line of input contains two integers, n and m ($1 \leq n, m \leq 50$), which are the height and width of the image. The next n lines will each contain exactly m characters, consisting only of '`L`', '`W`' and '`C`', as explained above.

Output

Print one line of output containing an integer denoting the minimum number of islands possible.

Examples

Sample Input	Sample Output
4 5 CCCCC CCCCC CCCCC CCCCC	0
3 2 LW CC WL	1

In the first example, the planet could be all water under the clouds.

In the second, the planet beneath the clouds could look like

```
LW
LL
WL
```

Problem G: Ghostbusters 2

In the 1984 Ghostbusters[2] movie, the protagonists use proton pack weapons that fire laser streams. This leads to the following memorable dialog between scientists Peter Venkman and Egon Spengler:

> **Spengler:** "There's something very important I forgot to tell you.
> **Venkman:** What?
> **Spengler:** Don't cross the streams.
> **Venkman:** Why?
> **Spengler:** It would be bad.
> **Venkman:** I'm fuzzy on the whole good/bad thing. What do you mean, "bad"?
> **Spengler:** Try to imagine all life as you know it stopping instantaneously and every molecule in your body exploding at the speed of light.
> **Venkman:** Right. That's bad. Okay. All right. Important safety tip"

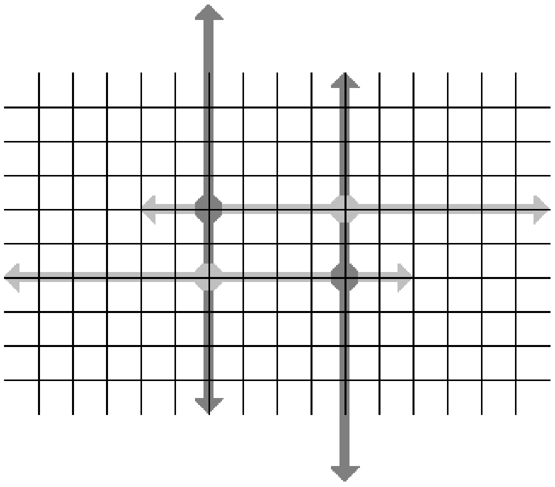
In the 30+ years since that time, there have been several technical advances in their weapons systems. They are currently trying out a new prototype:

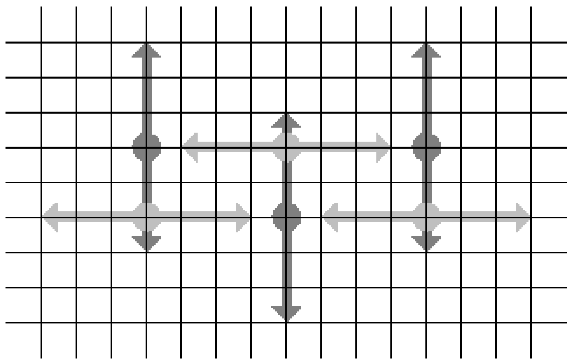
- The streams have been polarized, firing either horizontally or vertically. There is no longer any danger if streams having opposite polarity cross each other. However, there will still be catastrophic results if two streams having the same orientation in any way touch each other.
- The streams do not affect living creatures. A stream may pass through one of the ghostbusters with no ill effects.
- A weapon now simultaneously fires its streams in opposite directions.

  More specifically, a weapon has an integer power P and when fired will reach locations P units to the left and P units to the right of the ghostbuster, if fired horizontally, or P units above and below the ghostbuster if fired vertically.

When stationed at their positions, the ghostbusters can communicate to decide who will fire horizontally and who will fire vertically. They will all use the same power value P and would like to use as much power as possible without causing a catastrophe.

[2]"Ghostbusters" is a registered trademark of Columbia Pictures Industries, Inc., and the dialog quoted on this page is © Columbia Pictures Industries, Inc.





As an example, the plot on the left shows a configuration of ghostbusters in which an arbitrarily large power value can be used, so long as the ghostbusters coordinate their orientations. The plot on the right shows a configuration in which there is an orientation to allow power level 3, but for which no orientation allows power level 4. (Notice that the streams of the bottom-left and bottom-right ghostbusters would touch if using power level 4 with that same orientation of streams.)

Input

The first line contains an integer N, such that $1 \leq N \leq 1000$, indicating the number of ghostbusters. Following that are N lines, each containing integers x and y which describe the location of one ghostbuster, such that $0 \leq x, y \leq 1{,}000{,}000$. No two ghostbusters are at the same location.

Output

If the ghostbusters can use arbitrarily large power without catastrophe, print `UNLIMITED`. Otherwise print the largest integer power value that may be safely used with appropriately chosen orientations.

Time Limit

Solutions to this problem must run in no more than 10 seconds.

Examples

Sample Input	Sample Output

4
5 4
9 4
5 6
9 6
	UNLIMITED

6
3 3
7 3
11 3
3 5
7 5
11 5
	3

6
0 0
1 0
2 0
0 1
1 1
2 1
	0

Problem H: Painting the Floodwall

The town of South Riverside has a long floodwall to protect the residents against occasional rising waters from the nearby Little Muddy river. It's very functional, but also a bit of an eyesore.

The town has decided to spruce it up by staging a competition for local artists to paint murals on sections of the wall. Artists have submitted applications for the contest, in which they have specified not only how long a section of wall they want to paint but also, based upon the surrounding scenery, where along the wall they would like to place their mural.

Obviously, the artists' work cannot overlap, so there is a possibility that not all artists' applications can be accepted. On the other hand, the town would like to see as much of the wall painted as possible.

Find the combination of artists whose applications can be accepted to maximize the amount of the wall painted without allowing any artists' work to overlap. A mural that starts at the same coordinate at which another mural stops is not considered to overlap.

Input

Input will begin with a line containing an integer N denoting the number of artists whose have submitted applications. $1 \leq N \leq 200{,}000$

This will be followed by N lines, each containing two integers x_0 and x_1, $0 \leq x_0 \leq x_1 \leq 10^{18}$, denoting a starting and ending position (inclusive) for a proposed mural.

Output

Print a single line containing the maximum total length of the fence that can be painted without allowing any two artists' work to overlap.

Example

Sample Input	Sample Output

3
100 200
190 210
200 300
	200

5
0 5
12 18
4 14
7 9
17 18
	13

The second example reflects a decision to accept the applications to paint portions $0 \ldots 5$, $7 \ldots 9$, and $12 \ldots 18$.