Problem A
Self-Assembly
Time Limit: 4 seconds

Automatic Chemical Manufacturing is experimenting with a process called self-assembly. In this process, molecules with natural affinity for each other are mixed together in a solution and allowed to spontaneously assemble themselves into larger structures. But there is one problem: sometimes molecules assemble themselves into a structure of unbounded size, which gums up the machinery.

You must write a program to decide whether a given collection of molecules can be assembled into a structure of unbounded size. You should make two simplifying assumptions: 1) the problem is restricted to two dimensions, and 2) each molecule in the collection is represented as a square. The four edges of the square represent the surfaces on which the molecule can connect to other compatible molecules.

In each test case, you will be given a set of molecule descriptions. Each type of molecule is described by four two-character *connector labels* that indicate how its edges can connect to the edges of other molecules. There are two types of connector labels:

- An uppercase letter $(A, \ldots, Z)$ followed by $+$ or $-$. Two edges are compatible if their labels have the same letter but different signs. For example, $A+$ is compatible with $A-$ but is not compatible with $A+$ or $B-$.
- Two zero digits 00. An edge with this label is not compatible with any edge (not even with another edge labeled 00).

Assume there is an unlimited supply of molecules of each type, which may be rotated and reflected. As the molecules assemble themselves into larger structures, the edges of two molecules may be adjacent to each other only if they are compatible. It is permitted for an edge, regardless of its connector label, to be connected to nothing (no adjacent molecule on that edge).

Figure G.1 shows an example of three molecule types and a structure of bounded size that can be assembled from them (other bounded structures are also possible with this set of molecules).

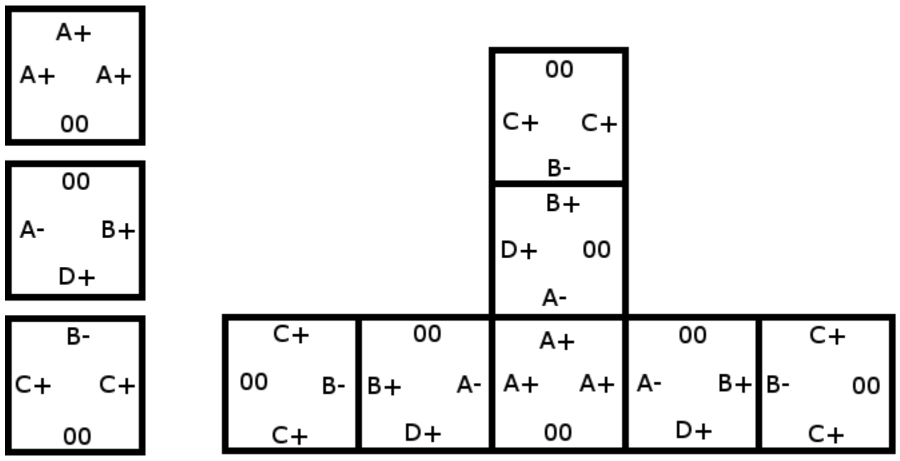

Figure A.1: Illustration of Sample Input 1.

Input

The input consists of a single test case. A test case has two lines. The first line contains an integer n ($1 \le n \le 40\,000$) indicating the number of molecule types. The second line contains n eight-character strings, each describing a single type of molecule, separated by single spaces. Each string consists of four two-character connector labels representing the four edges of the molecule in clockwise order.

Output

Display the word `unbounded` if the set of molecule types can generate a structure of unbounded size. Otherwise, display the word `bounded`.

Sample Input 1	Sample Output 1
3 A+00A+A+ 00B+D+A- B-C+00C+	bounded

Sample Input 2	Sample Output 2
1 K+K-Q+Q-	unbounded

Problem B
Curvy Little Bottles
Time Limit: 1 second

In her bike rides around Ekaterinburg, Jill happened upon a shop that sold interesting glass bottles. She thought it might make an interesting project to use such bottles for measuring liquids, but this would require placing markings on the bottles to indicate various volumes. Where should those volume marks be placed?

Jill formalized the problem as follows. Assume a bottle is formed by revolving a shape that is the same as the graph of a polynomial P between $x = x_{\text{low}}$ and $x = x_{\text{high}}$ around the x-axis. Thus the x-axis is coincident with a vertical line through the center of the bottle. The bottom of the bottle is formed by a solid circular region at $x = x_{\text{low}}$, and the top of the bottle, at $x = x_{\text{high}}$, is left open.

The first sample input represents a bottle formed using the simple polynomial $4 - 0.25x$, with $x_{\text{low}} = 0$ and $x_{\text{high}} = 12$. The bottom of this bottle is a circle with a radius of 4, and the opening at the top is a circle with a radius of 1. The height of this bottle is 12. Volume markings are in increments of 25.

Given a polynomial P, x_{low}, x_{high}, and the volume increment between successive marks on the bottle, compute the distances up from x_{low} for the marks at successive volume increments. A mark cannot be made past the top of the bottle, and no more than the first 8 increments should be marked. Assume the value of P is greater than zero everywhere between x_{low} and x_{high}.

Input

The input consists of a single test case. A test case has three lines of bottle data:

- Line 1: n, the degree of the polynomial (an integer satisfying $0 \leq n \leq 10$).
- Line 2: $a_0, a_1, \ldots, a_n$, the real coefficients of the polynomial P defining the bottle's shape, where a_0 is the constant term, a_1 is the coefficient of x^1, …, and a_n is the coefficient of x^n. For each i, $-100 \leq a_i \leq 100$, and $a_n \neq 0$.
- Line 3:
  - x_{low} and x_{high}, the real-valued boundaries of the bottle ($-100 \leq x_{\text{low}} < x_{\text{high}} \leq 100$ and $x_{\text{high}} - x_{\text{low}} > 0.1$).
  - inc, an integer which is the volume increment before each successive mark on the bottle ($1 \leq inc \leq 500$).

Output

Display the volume of the full bottle on one line. On a second line, display the increasing sequence of no more than 8 successive distances up from the bottom of the bottle for the volume markings. All volumes and height marks should have absolute error of at most 10^{-3}. If the bottle does not have a volume that allows at least one mark, display the phrase `insufficient volume`. No test case will result in a mark within 0.01 from the top of the bottle. The volume of the bottle will not exceed $1\,000$. All distances for marks on a bottle differ by at least 0.04.

Sample Input 1

```
1
4.0 -0.25
0.0 12.0 25
```

Sample Output 1

```
263.8938
0.5137 1.0639 1.6579 2.3058 3.0215 3.8262 4.7543 5.8677
```

Sample Input 2

```
0
1.0
0.0 10.0 10
```

Sample Output 2

```
31.4159
3.1831 6.3662 9.5493
```

Sample Input 3

```
0
1.7841241161782
5.0 10.0 20
```

Sample Output 3

```
50.0000
2.0000 4.0000
```

Sample Input 4

```
1
4.0 -0.25
0.0 12.0 300
```

Sample Output 4

```
263.8938
insufficient volume
```

Problem C
Fibonacci Words
Time Limit: 1 second

The Fibonacci word sequence of bit strings is defined as:

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n \geq 2 \end{cases}$$

Here $+$ denotes concatenation of strings. The first few elements are:

n	$F(n)$
0	0
1	1
2	10
3	101
4	10110
5	10110101
6	1011010110110
7	101101011011010110101
8	1011010110110101101011011010110110
9	1011010110110101101011011010110110101101011011010110101

Given a bit pattern p and a number n, how often does p occur in $F(n)$?

Input

The input consists of at most 100 test cases. The first line of each test case contains the integer n ($0 \leq n \leq 100$). The second line contains the bit pattern p. The pattern p is nonempty and has a length of at most 100 000 characters.

Output

For each test case, print the number of occurrences of the bit pattern p in $F(n)$. Occurrences may overlap. The number of occurrences will be less than 2^{63}.

Sample Input 1

```
6
10
7
10
6
01
6
101
96
10110101101101
```

Sample Output 1

```
5
8
4
4
7540113804746346428
```

This page is intentionally left blank.

Problem D
Low Power
Time Limit: 3 seconds

You are building advanced chips for machines. Making the chips is easy, but the power supply turns out to be an issue since the available batteries have varied power outputs.

Consider the problem of n machines, each with two chips, where each chip is powered by k batteries. Surprisingly, it does not matter how much power each chip gets, but a machine works best when its two chips have power outputs as close as possible. The power output of a chip is simply the smallest power output of its k batteries.

You have a stockpile of $2nk$ batteries that you want to assign to the chips. It might not be possible to allocate the batteries so that in every machine both chips have equal power outputs, but you want to allocate them so that the differences are as small as possible. To be precise, you want to tell your customers that in all machines the difference of power outputs of the two chips is at most d, and you want to make d as small as possible. To do this you must determine an optimal allocation of the batteries to the machines.

Consider Sample Input 1. There are 2 machines, each requiring 3 batteries per chip, and a supply of batteries with power outputs $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$. You can, for instance, assign the batteries with power outputs $1, 3, 5$ to one chip, those with power $2, 4, 12$ to the other chip of the same machine, those with power $6, 8, 9$ to the third chip, and those with power $7, 10, 11$ to the fourth. The power outputs of the chips are $1, 2, 6$, and 7, respectively, and the difference between power outputs is 1 in both machines. Note that there are many other ways to achieve this result.

Input

The input consists of a single test case. A test case has two lines. The first line contains two positive integers: the number of machines n and the number of batteries per chip k ($2nk \leq 10^6$). The second line contains $2nk$ integers p_i specifying the power outputs of the batteries ($1 \leq p_i \leq 10^9$).

Output

Display the smallest number d such that you can allocate the batteries so that the difference of power outputs of the two chips in each machine is at most d.

Sample Input 1	Sample Output 1
2 3 1 2 3 4 5 6 7 8 9 10 11 12	1

Sample Input 2	Sample Output 2
2 2 3 1 3 3 3 3 3 3	2

This page is intentionally left blank.

Problem E
Матрёшка
Time Limit: 5 seconds

Matryoshkas are sets of traditional Russian wooden dolls of decreasing size placed one inside the other. A matryoshka doll can be opened to reveal a smaller figure of the same sort inside, which has, in turn, another figure inside, and so on.

The Russian Matryoshka Museum recently exhibited a collection of similarly designed matryoshka sets, differing only in the number of nested dolls in each set. Unfortunately, some over-zealous (and obviously unsupervised) children separated these sets, placing all the individual dolls in a row. There are n dolls in the row, each with an integer size. You need to reassemble the matryoshka sets, knowing neither the number of sets nor the number of dolls in each set. You know only that every complete set consists of dolls with consecutive sizes from 1 to some number m, which may vary between the different sets.

Picture from Wikimedia Commons

When reassembling the sets, you must follow these rules:

- You can put a doll or a nested group of dolls only inside a larger doll.
- You can combine two groups of dolls only if they are adjacent in the row.
- Once a doll becomes a member of a group, it cannot be transferred to another group or permanently separated from the group. It can be temporarily separated only when combining two groups.

Your time is valuable, and you want to do this reassembly process as quickly as possible. The only time-consuming part of this task is opening and subsequently closing a doll, so you want to minimize how often you do this. For example, the minimum number of openings (and subsequent closings) when combining group [1, 2, 6] with the group [4] is two, since you have to open the dolls with sizes 6 and 4. When combining group [1, 2, 5] with the group [3, 4], you need to perform three openings.

Write a program to calculate the minimum number of openings required to combine all disassembled matryoshka sets.

Input

The input consists of a single test case. A test case has two lines. The first line contains one integer n ($1 \le n \le 500$) representing the number of individual dolls in the row. The second line contains n positive integers specifying the sizes of the dolls in the order they appear in the row. Each size is between 1 and 500 inclusive.

Output

Display the minimum number of openings required when reassembling the matryoshka sets. If reassembling cannot be done (some of the kids might have been excessively zealous and taken some dolls), display the word `impossible`.

Sample Input 1

```
7
1 2 3 2 4 1 3
```

Sample Output 1

```
7
```

Sample Input 2

```
7
1 2 1 2 4 3 3
```

Sample Output 2

```
impossible
```

Problem F
Ragged Right
Time Limit: 1 second

Word wrapping is the task of deciding how to break a paragraph of text into lines. For aesthetic reasons, all the lines except the last should be about the same length. For example, the text on the left looks less ragged than the text on the right:

```
This is a            This
paragraph            is a paragraph
of text.             of text.
```

Your job is to compute a raggedness value for an arbitrary paragraph of text. Measure raggedness in a way similar to the TEX typesetting system. Let n be the length, measured in characters, of the longest line of the paragraph. If some other line contains only m characters, then charge a penalty score of $(n-m)^2$ for that line. The raggedness is the sum of the penalty scores for every line except the last one.

Input

The input consists of a single test case. A test case is a single paragraph of text containing between 1 and 100 lines. Each line of the paragraph contains between 1 and 80 characters (letters, punctuation characters, decimal digits and spaces). No line starts or ends with spaces.

Output

Display a single integer, which is the raggedness score for the paragraph.

Sample Input 1

```
some blocks
of text line up
well on the right,
but
some don't.
```

Sample Output 1

```
283
```

Sample Input 2

```
this line is short
this one is a bit longer
and this is the longest of all.
```

Sample Output 2

```
218
```

This page is intentionally left blank.

Problem G
Self-Assembly 2
Time Limit: 4 seconds

Automatic Chemical Manufacturing is experimenting with a process called self-assembly. In this process, molecules with natural affinity for each other are mixed together in a solution and allowed to spontaneously assemble themselves into larger structures. But there is one problem: sometimes molecules assemble themselves into a structure of unbounded size, which gums up the machinery.

You must write a program to decide whether a given collection of molecules can be assembled into a structure of unbounded size. You should make two simplifying assumptions: 1) the problem is restricted to two dimensions, and 2) each molecule in the collection is represented as a square. The four edges of the square represent the surfaces on which the molecule can connect to other compatible molecules.

In each test case, you will be given a set of molecule descriptions. Each type of molecule is described by four two-character *connector labels* that indicate how its edges can connect to the edges of other molecules. There are two types of connector labels:

- An uppercase letter ($A, \ldots, Z$) followed by $+$ or $-$. Two edges are compatible if their labels have the same letter but different signs. For example, $A+$ is compatible with $A-$ but is not compatible with $A+$ or $B-$.
- Two zero digits 00. An edge with this label is not compatible with any edge (not even with another edge labeled 00).

Assume there is an unlimited supply of molecules of each type, which may be rotated and reflected. As the molecules assemble themselves into larger structures, the edges of two molecules may be adjacent to each other only if they are compatible. It is permitted for an edge, regardless of its connector label, to be connected to nothing (no adjacent molecule on that edge).

Figure G.1 shows an example of three molecule types and a structure of bounded size that can be assembled from them (other bounded structures are also possible with this set of molecules).

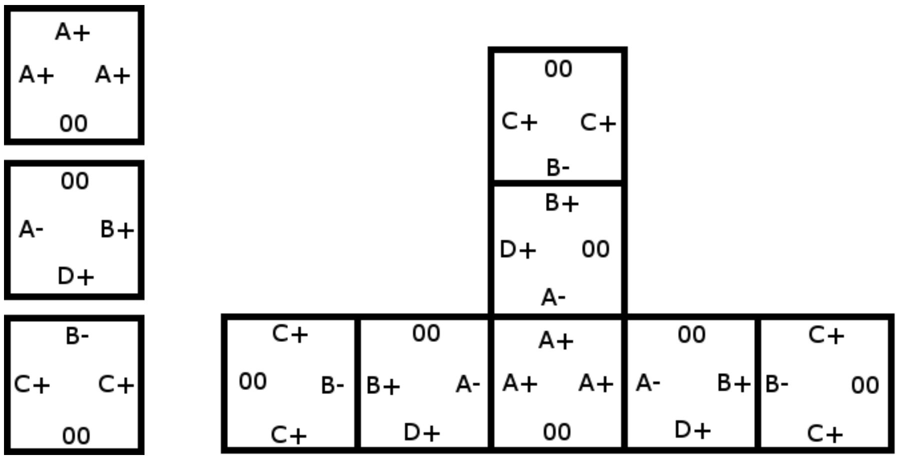

Figure G.1: Illustration of Sample Input 1.

Input

The input consists of a single test case. A test case has two lines. The first line contains an integer n ($1 \leq n \leq 40\,000$) indicating the number of molecule types. The second line contains n eight-character strings, each describing a single type of molecule, separated by single spaces. Each string consists of four two-character connector labels representing the four edges of the molecule in clockwise order.

Output

Display the word `unbounded` if the set of molecule types can generate a structure of unbounded size. Otherwise, display the word `bounded`.

Sample Input 1

```
3
A+00A+A+ 00B+D+A- B-C+00C+
```

Sample Output 1

```
bounded
```

Sample Input 2

```
1
K+K-Q+Q-
```

Sample Output 2

```
unbounded
```

Problem H
Curvy Little Bottles 2
Time Limit: 1 second

In her bike rides around Ekaterinburg, Jill happened upon a shop that sold interesting glass bottles. She thought it might make an interesting project to use such bottles for measuring liquids, but this would require placing markings on the bottles to indicate various volumes. Where should those volume marks be placed?

Jill formalized the problem as follows. Assume a bottle is formed by revolving a shape that is the same as the graph of a polynomial P between $x = x_{\text{low}}$ and $x = x_{\text{high}}$ around the x-axis. Thus the x-axis is coincident with a vertical line through the center of the bottle. The bottom of the bottle is formed by a solid circular region at $x = x_{\text{low}}$, and the top of the bottle, at $x = x_{\text{high}}$, is left open.

The first sample input represents a bottle formed using the simple polynomial $4 - 0.25x$, with $x_{\text{low}} = 0$ and $x_{\text{high}} = 12$. The bottom of this bottle is a circle with a radius of 4, and the opening at the top is a circle with a radius of 1. The height of this bottle is 12. Volume markings are in increments of 25.

Given a polynomial P, x_{low}, x_{high}, and the volume increment between successive marks on the bottle, compute the distances up from x_{low} for the marks at successive volume increments. A mark cannot be made past the top of the bottle, and no more than the first 8 increments should be marked. Assume the value of P is greater than zero everywhere between x_{low} and x_{high}.

Input

The input consists of a single test case. A test case has three lines of bottle data:

- Line 1: n, the degree of the polynomial (an integer satisfying $0 \le n \le 10$).
- Line 2: $a_0, a_1, \ldots, a_n$, the real coefficients of the polynomial P defining the bottle's shape, where a_0 is the constant term, a_1 is the coefficient of x^1, …, and a_n is the coefficient of x^n. For each i, $-100 \le a_i \le 100$, and $a_n \ne 0$.
- Line 3:
  - x_{low} and x_{high}, the real-valued boundaries of the bottle ($-100 \le x_{\text{low}} < x_{\text{high}} \le 100$ and $x_{\text{high}} - x_{\text{low}} > 0.1$).
  - inc, an integer which is the volume increment before each successive mark on the bottle ($1 \le inc \le 500$).

Output

Display the volume of the full bottle on one line. On a second line, display the increasing sequence of no more than 8 successive distances up from the bottom of the bottle for the volume markings. All volumes and height marks should have absolute error of at most 10^{-3}. If the bottle does not have a volume that allows at least one mark, display the phrase `insufficient volume`. No test case will result in a mark within 0.01 from the top of the bottle. The volume of the bottle will not exceed $1\,000$. All distances for marks on a bottle differ by at least 0.04.

Sample Input 1

```
1
4.0 -0.25
0.0 12.0 25
```

Sample Output 1

```
263.8938
0.5137 1.0639 1.6579 2.3058 3.0215 3.8262 4.7543 5.8677
```

Sample Input 2

```
0
1.0
0.0 10.0 10
```

Sample Output 2

```
31.4159
3.1831 6.3662 9.5493
```

Sample Input 3

```
0
1.7841241161782
5.0 10.0 20
```

Sample Output 3

```
50.0000
2.0000 4.0000
```

Sample Input 4

```
1
4.0 -0.25
0.0 12.0 300
```

Sample Output 4

```
263.8938
insufficient volume
```

Problem I
Fibonacci Words 2
Time Limit: 1 second

The Fibonacci word sequence of bit strings is defined as:

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n \geq 2 \end{cases}$$

Here $+$ denotes concatenation of strings. The first few elements are:

n	$F(n)$
0	0
1	1
2	10
3	101
4	10110
5	10110101
6	1011010110110
7	101101011011010110101
8	1011010110110101101011011010110110
9	1011010110110101101011011010110110101101011011010110101

Given a bit pattern p and a number n, how often does p occur in $F(n)$?

Input

The input consists of at most 100 test cases. The first line of each test case contains the integer n ($0 \leq n \leq 100$). The second line contains the bit pattern p. The pattern p is nonempty and has a length of at most 100 000 characters.

Output

For each test case, print the number of occurrences of the bit pattern p in $F(n)$. Occurrences may overlap. The number of occurrences will be less than 2^{63}.

Sample Input 1

```
6
10
7
10
6
01
6
101
96
10110101101101
```

Sample Output 1

```
5
8
4
4
7540113804746346428
```

This page is intentionally left blank.

Problem J
Low Power 2
Time Limit: 3 seconds

You are building advanced chips for machines. Making the chips is easy, but the power supply turns out to be an issue since the available batteries have varied power outputs.

Consider the problem of n machines, each with two chips, where each chip is powered by k batteries. Surprisingly, it does not matter how much power each chip gets, but a machine works best when its two chips have power outputs as close as possible. The power output of a chip is simply the smallest power output of its k batteries.

You have a stockpile of $2nk$ batteries that you want to assign to the chips. It might not be possible to allocate the batteries so that in every machine both chips have equal power outputs, but you want to allocate them so that the differences are as small as possible. To be precise, you want to tell your customers that in all machines the difference of power outputs of the two chips is at most d, and you want to make d as small as possible. To do this you must determine an optimal allocation of the batteries to the machines.

Consider Sample Input 1. There are 2 machines, each requiring 3 batteries per chip, and a supply of batteries with power outputs $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$. You can, for instance, assign the batteries with power outputs $1, 3, 5$ to one chip, those with power $2, 4, 12$ to the other chip of the same machine, those with power $6, 8, 9$ to the third chip, and those with power $7, 10, 11$ to the fourth. The power outputs of the chips are $1, 2, 6$, and 7, respectively, and the difference between power outputs is 1 in both machines. Note that there are many other ways to achieve this result.

Input

The input consists of a single test case. A test case has two lines. The first line contains two positive integers: the number of machines n and the number of batteries per chip k ($2nk \leq 10^6$). The second line contains $2nk$ integers p_i specifying the power outputs of the batteries ($1 \leq p_i \leq 10^9$).

Output

Display the smallest number d such that you can allocate the batteries so that the difference of power outputs of the two chips in each machine is at most d.

Sample Input 1	Sample Output 1
2 3 1 2 3 4 5 6 7 8 9 10 11 12	1

Sample Input 2	Sample Output 2
2 2 3 1 3 3 3 3 3 3	2

Problem K
Матрёшка 2
Time Limit: 5 seconds

Matryoshkas are sets of traditional Russian wooden dolls of decreasing size placed one inside the other. A matryoshka doll can be opened to reveal a smaller figure of the same sort inside, which has, in turn, another figure inside, and so on.

The Russian Matryoshka Museum recently exhibited a collection of similarly designed matryoshka sets, differing only in the number of nested dolls in each set. Unfortunately, some over-zealous (and obviously unsupervised) children separated these sets, placing all the individual dolls in a row. There are n dolls in the row, each with an integer size. You need to reassemble the matryoshka sets, knowing neither the number of sets nor the number of dolls in each set. You know only that every complete set consists of dolls with consecutive sizes from 1 to some number m, which may vary between the different sets.

Picture from Wikimedia Commons

When reassembling the sets, you must follow these rules:

- You can put a doll or a nested group of dolls only inside a larger doll.
- You can combine two groups of dolls only if they are adjacent in the row.
- Once a doll becomes a member of a group, it cannot be transferred to another group or permanently separated from the group. It can be temporarily separated only when combining two groups.

Your time is valuable, and you want to do this reassembly process as quickly as possible. The only time-consuming part of this task is opening and subsequently closing a doll, so you want to minimize how often you do this. For example, the minimum number of openings (and subsequent closings) when combining group [1, 2, 6] with the group [4] is two, since you have to open the dolls with sizes 6 and 4. When combining group [1, 2, 5] with the group [3, 4], you need to perform three openings.

Write a program to calculate the minimum number of openings required to combine all disassembled matryoshka sets.

Input

The input consists of a single test case. A test case has two lines. The first line contains one integer n ($1 \le n \le 500$) representing the number of individual dolls in the row. The second line contains n positive integers specifying the sizes of the dolls in the order they appear in the row. Each size is between 1 and 500 inclusive.

Output

Display the minimum number of openings required when reassembling the matryoshka sets. If reassembling cannot be done (some of the kids might have been excessively zealous and taken some dolls), display the word `impossible`.

Sample Input 1

```
7
1 2 3 2 4 1 3
```

Sample Output 1

```
7
```

Sample Input 2

```
7
1 2 1 2 4 3 3
```

Sample Output 2

```
impossible
```

Problem L
Ragged Right 2
Time Limit: 1 second

Word wrapping is the task of deciding how to break a paragraph of text into lines. For aesthetic reasons, all the lines except the last should be about the same length. For example, the text on the left looks less ragged than the text on the right:

```
This is a                   This
paragraph                   is a paragraph
of text.                    of text.
```

Your job is to compute a raggedness value for an arbitrary paragraph of text. Measure raggedness in a way similar to the TEX typesetting system. Let n be the length, measured in characters, of the longest line of the paragraph. If some other line contains only m characters, then charge a penalty score of $(n-m)^2$ for that line. The raggedness is the sum of the penalty scores for every line except the last one.

Input

The input consists of a single test case. A test case is a single paragraph of text containing between 1 and 100 lines. Each line of the paragraph contains between 1 and 80 characters (letters, punctuation characters, decimal digits and spaces). No line starts or ends with spaces.

Output

Display a single integer, which is the raggedness score for the paragraph.

Sample Input 1	Sample Output 1
some blocks of text line up well on the right, but some don't.	283

Sample Input 2	Sample Output 2
this line is short this one is a bit longer and this is the longest of all.	218