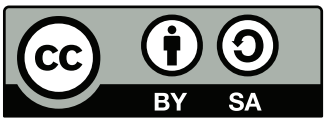
# BAPC 2025

*The 2025 Benelux Algorithm Programming Contest*


## Problems



# A Accidental Arithmetic

Time limit: 1s

It is the 25th of January, 2025. You are working on your final report for the statistics class. However, when you enter the calculations into your trusted Brittle Arithmetic Portable Calculator, you notice that the [+] and [−] buttons have malfunctioned. Whenever you press a numeric button (a button from [0] through [9]), an additional button press may be registered immediately after: either the [+] button, or the [−] button, or neither (but never both). "Ugh," you think to yourself, "why did I not buy a Backup Arithmetic Portable Calculator..." But then you get an idea to save yourself from getting a low grade.

Your trusted Brittle Arithmetic Portable Calculator. Drawing by Freek Henstra

After some experimentation, you statistically determined that pressing a numeric button results in an additional [+] input 45% of the time, and the same holds for the [−] button (and never both at the same time). You decide that you should write your report about your findings. In addition to statistical analysis, the teacher requires you to perform some probabilistic predictions based on a statistical model. You decide to investigate what your calculator does when you try to simply input a natural number.

Given a natural number $n$, you consider entering the number into the calculator by pressing the numeric buttons corresponding to the standard base 10 representation of $n$. The calculator may register a [+] or [−] button press in between the digits of $n$, which results in an arithmetic expression. You wonder what happens if you press [=] to evaluate the expression. Luckily, if your expression ends with a [+] or [−], the calculator ignores this, such that this evaluation always results in an integer. To finish your report, you decide to write a program that determines the expected value of this result for any given value of $n$.

## Input

The input consists of:

- One line with an integer $n$ ($0 \leq n < 10^{1000}$), the number that you will enter into the calculator.

## Output

Output the expected value of the result.

Your answer should have an absolute or relative error of at most $10^{-6}$.

| Sample Input 1 | Sample Output 1 |
|---|---|
| 12345 | 5.4321 |

| **Sample Input 2** | **Sample Output 2** |
|---|---|
| 777777 | 42 |

**Input 3**

```
31415926535897932384626433832795028841971693993751
```

**Output 3**

```
141.5189
```

## B Boggle Sort

Time limit: 1s

It is the 25th of February, 2025. You have enjoyed another spirited evening of *Boggle* with your friends. After everybody left, you have thoroughly cleaned the apartment. All that is left is to bring the Boggle tray in order. You start to wonder: would it be possible to bring the Boggle tray in alphabetic order, without swapping any dice, but only by rotating them?

A tray of Boggle dice, out of order. CC-BY-2.0 by Rich Brooks on Wikimedia Commons

The Boggle tray consists of 16 six-sided dice. Each die is labelled with a letter from the English alphabet on each face. A single die contains a face labelled "Qu". No letter appears 4 or more times on the same die. By turning a die once, you can move any of the sideways-facing letters up. Turning a die twice moves the downwards-facing letter up.

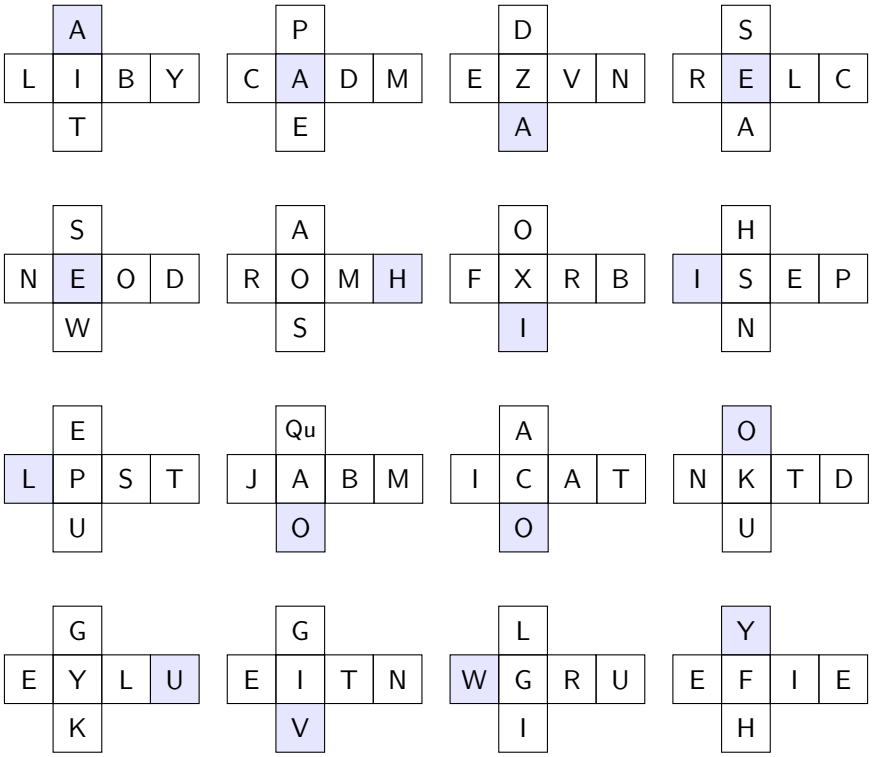



Figure B.1: Visualization of the first sample input. The 16 Boggle dice are shown in reading order. For each die, the face in the center of the cross ("I" in the first die) is upwards-facing and the face on the far right ("Y" in the first die) is downwards-facing. The shaded die faces describe an optimal solution requiring 15 turns.

Bring the tray into alphabetically nondecreasing order, using standard reading directions (left-to-right, top-to-bottom), using as few turns as possible. Letter case plays no role and the two-letter face is treated as "Q" followed by "U", so "QuU" is sorted but "QuT" is not.

## Input

The input consists of:

- One line with 16 letters, describing the currently upwards-facing faces of each die.
- Four lines with 16 letters, describing the currently sideways-facing faces of each die.
- One line with 16 letters, describing the downwards-facing faces of each die.

In each line, the $i$th letter describes the $i$th die for $1 \leq i \leq 16$.
All letters are English uppercase letters (`A`–`Z`).
The letter "`Q`" stands for the two-letter face "Qu" and appears exactly once in the input.
No letter appears 4 or more times on the same die.

## Output

If it is possible to bring the tops of the dice into alphabetic order, output the minimum number of turns needed to do so. Otherwise, output "`impossible`".

| **Sample Input 1** | **Sample Output 1** |
|---|---|
| IAZEEOXSPACKYIGF<br>APDSSAOHEQAOGGLY<br>LCERNRFILJINEEWE<br>BDVLOMRESBATLTRI<br>TEAAWSINUOOUKVIH<br>YMNCDHBPTMTDUNUE | 15 |

| **Sample Input 2** | **Sample Output 2** |
|---|---|
| EXFETDMNMGDBRSRM<br>TIEGINOVRETACNUA<br>PRYKASAEATNTSHID<br>SOHUOEJDHVKYLPLC<br>UFIYAWBZONUIEIWE<br>LBELCOQASIOLAEGP | impossible |

# C Coherency

Time limit: 8s

The miniatures are typically hand-painted by the players.

It is the 25th of March, 40025 CE in the world of *Battle Axe Player Clash 40,000* (BAPC40K). This futuristic table-top miniatures wargame is played with endearing figurines called *models*, each of which is placed on a circular *base*. The models are placed on a 100 km × 100 km gaming board. A collection of such models forms a *coherent unit* if between any pair of models there is an unbroken chain of models that have a Euclidean distance of at most two inches[1] between the edges of their bases. Moreover, if the unit contains seven models or more, each model must be within two inches of at least *two* other models. Given the positions of a collection of models with varying base diameters, determine whether they form a single coherent unit.

One can prove that for any valid input for this problem, if the diameters of the circular bases differ from the given diameter by at most $10^{-5}$ mm, the coherency of a unit of models does not change.

## Input

The input consists of:

- One line with an integer $n$ ($2 \leq n \leq 2 \cdot 10^5$), the number of models.
- $n$ lines, each with three integers $x$, $y$, and $d$ ($0 \leq x, y \leq 10^8$, $d \in \{25, 28, 32, 40, 50, 65, 80, 90, 100, 130, 160\}$), describing a model that has its center coordinates at $(x, y)$ and a base diameter of $d$, all given in millimeters.

Each model (including the base) fits on the gaming board.

It is guaranteed that no two models are overlapping, but the models can touch.

## Output

If the $n$ models form a single coherent unit, output "`yes`". Otherwise, output "`no`".

| Sample Input 1 | Sample Output 1 |
|---|---|
| 2<br>13 13 25<br>88 13 25 | yes |

| Sample Input 2 | Sample Output 2 |
|---|---|
| 2<br>13 13 25<br>89 13 25 | no |

[1] Recall that an inch equals 25.4 mm.

| **Sample Input 3** | **Sample Output 3** |
|---|---|
| 7<br>1255 1120 65<br>1204 1226 160<br>1090 1252 65<br>998 1179 160<br>998 1061 65<br>1090 988 160<br>1204 1014 65 | no |

| **Sample Input 4** | **Sample Output 4** |
|---|---|
| 7<br>1066 910 130<br>1007 1032 130<br>875 1062 130<br>770 978 130<br>770 843 130<br>875 758 130<br>1007 788 130 | yes |

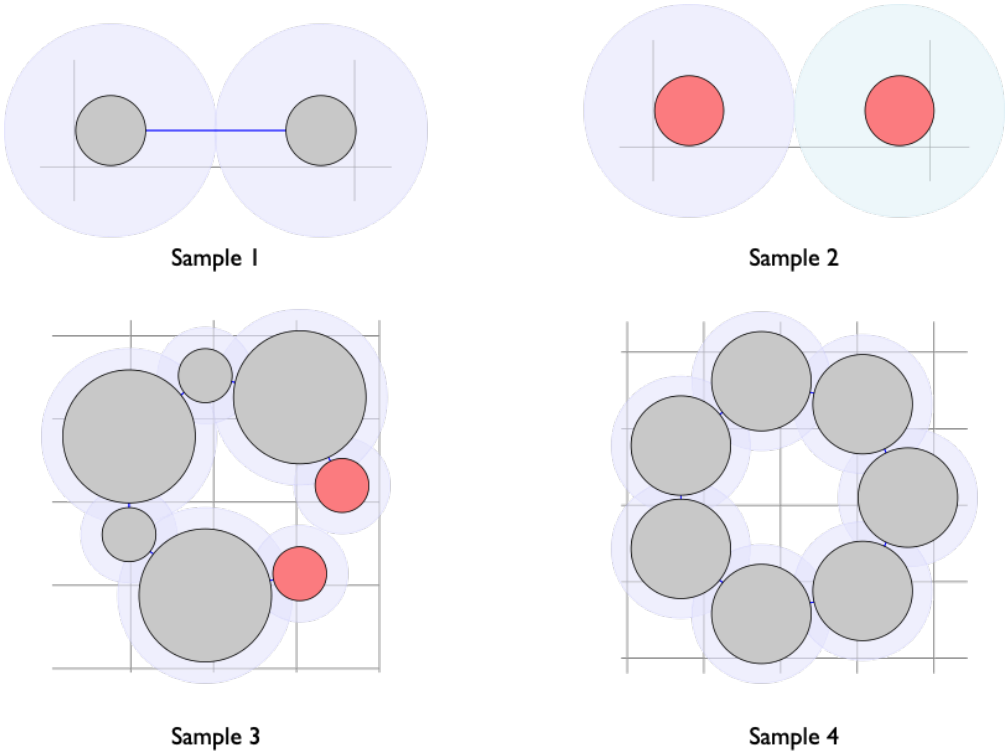



Figure C.1: Illustration of the samples. Samples 1 and 4 are coherent. Sample 2 is not coherent, because the two models are too far away. Sample 3 is not coherent, because not all models are within two inches of two other models.

# D Duo Detection

Time limit: 10s

It is the 25th of April, 1825. English inventer William Fothergill Cooke and English scientist Charles Wheatstone are working on an early prototype of an electrical telegraph system.[2] This system consists of a transmitter and a receiver. The transmitter can send messages to the receiver, which we model as a sequence of positive integers. However, the prototype is far from flawless, so quite often, an integer that is transmitted is not actually received. Integers never get altered through the connection, though, so all received integers are guaranteed to have been transmitted.

The Cooke and Wheatstone electric telegraph can transmit letters with the use of 5 needles which point to the transmitted letters. CC BY-SA 4.0 by Geni on Wikimedia Commons

To remedy this flaw, Cooke and Wheatstone came up with an error correction code. Together, they agreed on a fixed list of $n$ possible messages, and assigned each message a list of positive integers. To send a message from the list, they simply send the corresponding list of integers. Sometimes, some integers may get lost or arrive in a different order, but if enough integers remain, the hope is that there is still only one possible message.

You work as a postman, so you see the innovation of communication technology as a direct risk to your job security. You decide to mess with Cooke and Wheatstone to delay the development of their telegraph system. You have learned about their error correction code, and you have secretly obtained a copy of their message list with all the lists of integers. You have figured out a way to forcefully interrupt the transmission of an integer, so you decide to use this to ruin their system. To avoid suspicion, you decide that at least two integers should remain uninterrupted when a message is sent. Thus, your goal is to determine whether any message can be made ambiguous by interrupting all but two of its integers. Then, whenever such a message will be sent, you can make it ambiguous by interrupting the appropriate integers.

## Input

The input consists of:

- One line with an integer $n$ ($2 \leq n \leq 50\,000$), the number of possible messages.
- $n$ lines (one for each message), each with an integer $k$ ($2 \leq k \leq 10^5$), the number of integers assigned to the message, followed by those $k$ integers $x$ ($1 \leq x \leq 10^9$). It is guaranteed that the $k$ integers in a message are pairwise distinct.

The total number of integers in all messages combined is at most $10^5$.

[2] One may point out that Cooke and Wheatstone actually did not release a telegraph design until 1837, but obviously, this just means you were successful in this solving this problem! :)

### Output

If a pair of distinct integers exists that are assigned to multiple messages, output these two integers in any order, followed by the 1-based indices of two of the messages they are assigned to. Otherwise, output "`impossible`".

If there are multiple valid solutions, you may output any one of them.

| **Sample Input 1** | **Sample Output 1** |
|---|---|
| 3<br>5 1 9 3 7 5<br>4 2 4 6 8<br>4 7 5 3 2 | 3 5 3 1 |

| **Sample Input 2** | **Sample Output 2** |
|---|---|
| 3<br>2 42 1337<br>2 42 123456789<br>2 1337 123456789 | impossible |

# E Excruciating Elevators

Time limit: 2s

It is the 25th of May, 3025. The EEMCS building at TU Delft has grown to $10^6$ floors beyond the ground floor! The floors are now numbered $0, 1, \ldots, 10^6$, but there are still only four elevators. Moreover, the elevators have malfunctioned and are currently turned off. As employee of the Building Ascension Plans Company, you are tasked to fix the elevators.

The EEMCS building, now a million floors tall.

You first have to order some components, which will take a month to arrive at the ground floor. After collecting the components, you must visit $n$ floors $f_1, \ldots, f_n$ in order. At each floor $f_i$, you must replace some component, which takes $t_i$ seconds. After replacing the final component, the elevators will be fixed.

The stairs have long been removed, since people got tired of walking up millions of flights of stairs. You thus have to use the elevators to travel between the floors. You can turn on the elevators, but when you turn one on, you can no longer turn it off. Once turned on, an elevator will move up and down between floors 0 and $10^6$ indefinitely. The elevators move at a speed of one floor every second without stopping, but you are agile enough to enter and exit.

You know exactly when the components will arrive. In the meantime, you can determine when to turn on each elevator. This timing determines the starting configuration of each elevator when the components arrive. Since you have a month, you can enforce any arbitrary starting configuration of the elevators. What is the minimum possible time to fix the elevators?

As an example, consider the first sample input. You can ensure one elevator starts at ground floor going up, and another starts at floor 750 000 going up. Entering the former elevator immediately, you arrive at floor 600 000 after 600 000 seconds. After 50 000 more seconds, you finish replacing the component at floor 600 000. In the meantime, the latter elevator reached the top floor after 250 000 seconds, when it started going down. Now, 400 000 seconds later, this elevator is at floor 600 000 going down. You can enter this elevator immediately and exit 200 000 seconds later at floor 400 000. Finally, after 150 000 seconds, you finish replacing the component at floor 400 000, and the elevators are fixed! The total time is $600\,000 + 50\,000 + 200\,000 + 150\,000 = 1\,000\,000$ seconds.

## Input

The input consists of:

- One line with an integer $n$ ($1 \leq n \leq 35$), the number of floors to visit.
- One line with $n$ integers $f_1, \ldots, f_n$ ($0 \leq f_i \leq 10^6$ for each $i$), the numbers of the floors you must visit.
- One line with $n$ integers $t_1, \ldots, t_n$ ($1 \leq t_i \leq 10^9$ for each $i$), the necessary time spent on each floor in seconds.

No two consecutive floors $f_i, f_{i+1}$ are equal, and $f_1$ is non-zero.

### Output

Output the minimum possible time in seconds to fix the elevators after the components have arrived and you start at the ground floor.

| **Sample Input 1** | **Sample Output 1** |
|---|---|
| 2<br>600000 400000<br>50000 150000 | 1000000 |

| **Sample Input 2** | **Sample Output 2** |
|---|---|
| 10<br>1 2 3 4 5 6 7 8 9 10<br>1 1 1 1 1 1 1 1 1 1 | 4000012 |

# F Faulty Connection

Time limit: 1s

It is the 25th of June, 1825. English inventer William Fothergill Cooke and you, English scientist Charles Wheatstone, are working on an early prototype of an electrical telegraph system.[3] This system consists of a transmitter and a receiver. The transmitter can send messages to the receiver, which we model as a sequence of positive integers. However, the prototype is far from flawless, so quite often, an integer that is transmitted is not actually received. Integers never get altered through the connection, though, so all received integers are guaranteed to have been transmitted.

The Cooke and Wheatstone electric telegraph can transmit letters with the use of 5 needles which point to the transmitted letters. CC BY-SA 4.0 by Geni on Wikimedia Commons

To remedy this flaw, Cooke and you are working on an error correction code. The idea is to agree on a fixed list of 600 possible messages, indexed 1 to 600, and assign each message a list of 30 positive integers of at most 1000. To send a message from the list, you simply send the corresponding list of integers. Sometimes, some integers may get lost or arrive in a different order, but if enough integers remain, the hope is that there is still only one possible message.

When trying this out back in April, you noticed that, with some bad luck, the connection can get very faulty. Sometimes only two integers remained, which can easily make a message ambiguous! You decide to investigate whether this issue can be resolved purely by improving the error correction code. To each of the 600 possible messages, you need to assign a list of 30 positive integers of at most 1000, such that receiving any two integers in any order from any one of the lists uniquely determines the corresponding message.

Your program will be run multiple times for each test case. In the first pass, your program will be given an index of a message to send, which your program should assign a list of 30 integers. In subsequent passes, your program will be given two of the 30 integers from the first pass in any order, which it should then decode to retrieve the original message.

Your submission may take up to 1 second for each pass.

A testing tool is provided to help you develop your solution.

[3] One may point out that Cooke and Wheatstone actually did not release a telegraph design until 1837, but obviously, this is due to the problem we are trying to solve here! :)

### Input

This is a *multi-pass* problem.

The input consists of:

- One line with the action your program needs to perform:
  - "`send`" if you need to send a message.
  - "`receive`" if you need to receive two integers and determine the corresponding message.
- If the action is "`send`", one line with an integer $k$ ($1 \leq k \leq 600$), the index of the message to send.
- If the action is "`receive`", one line with two distinct integers $a$ and $b$ ($1 \leq a, b \leq 1000$) from the output of your program in the first pass, in any order.

### Output

If the action is "`send`", output 30 distinct integers $x$ ($1 \leq x \leq 1000$), assigning this list of integers to the message with index $k$.

If the action is "`receive`", output the index $k$ of the corresponding message.

### Note

The displayed sample output of a "`send`" action uses line wrapping for display purposes. Whitespace in the output is treated as usual.

**Sample Case 1**

| **Input** | **Pass 1** | **Output** |
|---|---|---|
| `send`<br>`42` | `814 734 58 792 286 974 893 735`<br>`538 498 916 163 226 32 160 659`<br>`980 994 775 334 44 492 276 983`<br>`398 885 179 888 755 121` | |

| **Input** | **Pass 2** | **Output** |
|---|---|---|
| `receive`<br>`286 58` | `42` | |

| **Input** | **Pass 3** | **Output** |
|---|---|---|
| `receive`<br>`163 492` | `42` | |

# G Garbage In, Garbage Out

Time limit: 2s

Similarly, in five hundred years, this is what artificially generated images will look like.

It is the 25th of July, 2525. Now that the usage of Large Language Models (LLMs) is so ubiquitous, it has become nearly impossible to find actual human-made articles on the internet and not get lost in all the gibberish produced by LLMs.

(Un)fortunately, LLM technology has regressed significantly. Neural networks have been trained on data that is mostly generated by older LLMs, which in turn were trained on even older recycled data. As a result, the output produced by most LLMs is a long string of lowercase letters, each chosen uniformly at random and independently of the others.

You decided to make a program to scavenge the internet in search of human-made articles. Your program must determine whether a given text is human-made or generated by an LLM.

A given text is guaranteed to be exactly one of the following:

- *Human-made*, in which case it is a fixed concatenation (without spaces) of words from a given word list.
- *Not human-made* and, therefore, generated by an LLM, in which case each character is chosen independently and uniformly at random.

## Input

The input consists of:

- One line with a string $s$, the given text to check.
- One line with an integer $n$, the number of words in the word list.
- $n$ lines, each with a string $w$ ($6 \leq |w| \leq 10$), the words in the word list. The words in the word list are distinct and fixed per test case.

All input strings only consist of English lowercase letters (`a-z`).

Your submission will be run on exactly 100 test cases, all of which will have $|s| = 3 \cdot 10^5$ and $n = 5000$. The samples are smaller and for illustration only.

For each test case where $s$ is human-made, $s$ is fixed and does not change between each of your submissions. For each test case where $s$ is generated by an LLM, each of your submissions will receive a new string $s$, generated from independently and uniformly picking random English lowercase letters (`a-z`).

### Output

If the given string was human-made, output "`yes`". Otherwise, if it was generated by an LLM, output "`no`".

### Input 1

```
balloncodingballoonacceptedchallengechallengecoding
5
accepted
balloon
challenge
coding
algorithms
```

### Output 1

```
yes
```

### Input 2

```
nlaiueakuyclocedxlwvxdbiifqjbinucjabzyaqakhohgympi
5
aaaaaaaaaa
bbbbbbbbbb
aababbb
aaabbbaaa
abaaba
```

### Output 2

```
no
```

### Input 3

```
aaaaaaaaaaabbbbbbbbbbaaaaaaaaaaaaaaabbbbbbbbbbbbbb
5
aaaaaaaaa
bbbbbbbbbb
aabbbbb
aaaaabbbb
bbbbba
```

### Output 3

```
yes
```

# H Homesick

Time limit: 2s

It is the 25th of August, 225 BCE. You are in charge of the annual road trip of the Backtracking-Averse Promenaders Club in Rome. Alas, you get homesick easily and would much rather stay at home. Therefore, your goal is to keep the road trip as short as possible. Traditionally, the road trip cannot backtrack along a road it just used – your friends would start to complain. Specifically, if you travel directly from site $x$ to site $y$, you cannot immediately go back from $y$ to $x$ along the same road.

Ancient Romans, promenading on Via Appia. Public Domain by Strafforello Gustavo on Wikimedia Commons

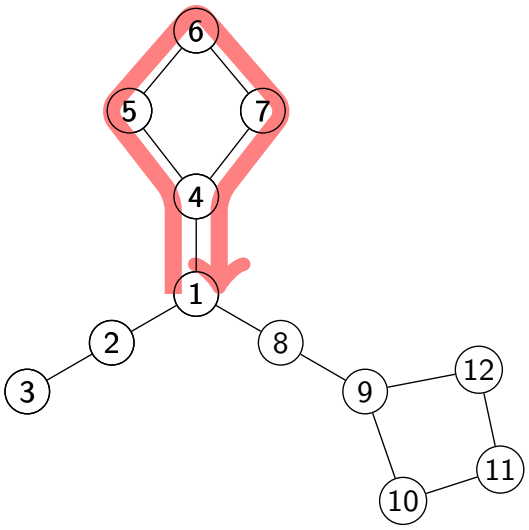



Figure H.1: Sample Input 2. The illustration shows a valid trip using six roads. The road connecting sites 1 and 4 is used twice.

You are given a list of sites to possibly visit and the roads connecting them. Find the road trip with the shortest length that would keep your friends happy.

The road trip must start at site 1, your home, and must visit at least one other site.

## Input

The input consists of:

- One line with two integers $n$ and $m$ ($2 \leq n \leq 10^5$, $1 \leq m \leq 2 \cdot 10^5$), the number of sites and the number of roads.
- $m$ lines, each with two integers $u$ and $v$ ($1 \leq u < v \leq n$), indicating there is a road between sites $u$ and $v$. Roads can be travelled in either direction. Each pair of sites is connected by at most one road.

### Output

If there is no road trip possible, output "`impossible`". Otherwise, output a your planned road trip, described by an integer $k$, the number of sites to visit on the road trip (including your home twice), followed by the $k$ sites, in the order of visiting them.

If there are multiple valid solutions, you may output any one of them.

**Sample Input 1**

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

**Sample Output 1**

```
4
1 3 2 1
```

**Sample Input 2**

```
12 13
1 2
2 3
1 4
4 5
5 6
6 7
4 7
1 8
8 9
9 10
10 11
11 12
9 12
```

**Sample Output 2**

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

**Sample Input 3**

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

**Sample Output 3**

```
impossible
```

**Sample Input 4**

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

**Sample Output 4**

```
impossible
```

# I Intermill Logistics

Time limit: 1s

It is the 25th of September, 1825. Having just completed a record-setting wheat harvest, you wonder what to do with all this wheat. With a stroke of genius, you decide to use all of this wheat to bake your favourite type of cookie: stroopwafels. Of course, all this wheat should be ground to flour first. Because you cannot wait to start baking, you want to do this as fast as possible, so you decide to contact all flour mills in the Netherlands to ask them for help.

One of the flour mills.

For each of these flour mills, you know how fast they can grind wheat to flour and how long it takes for a shipment to travel to and from the mill. You have enough grain ships available that can transport wheat to these mills, and bring the flour back from the mills. Dividing the wheat optimally between these mills, how long does it take until you have received back all the wheat?

As an example, consider the first sample case. To divide the wheat optimally between the three mills, you ship 400 kilograms to the first, 120 to the second, and 480 to the third. The first mill requires 5 hours to grind its wheat, the second requires 1 hour, and the third requires 3 hours. Combined with the shipping time to and from each mill, you receive all wheat back after exactly 11 hours.

In the second sample case, we send all of the wheat to the first mill. This mill can grind all 100 kilograms of wheat in 1 hour, which together with the 2 hours for shipping back and forth results in a total of 3 hours. As the shipping for the second mill would already take 4 hours, it is optimal to only use the first mill.

## Input

The input consists of:

- One line with two integers $n$ and $w$ ($1 \leq n \leq 10^5$, $1 \leq w \leq 10^9$), the number of mills and the amount of wheat you have, in kilograms.
- $n$ lines, each with two integers $p$ and $t$ ($1 \leq p, t \leq 10^9$), describing a mill that can process $p$ kilograms of wheat per hour, located $t$ hours away.

## Output

Output the number of hours until you have received back all the wheat, when dividing the wheat optimally between the mills.

Your answer should have an absolute or relative error of at most $10^{-6}$.

**Sample Input 1**

```
3 1000
80 3
120 5
160 4
```

**Sample Output 1**

```
11
```

**Sample Input 2**

```
2 100
100 1
500 2
```

**Sample Output 2**

```
3
```

**Sample Input 3**

```
3 7
1 1
1 1
1 1
```

**Sample Output 3**

```
4.3333333
```

# J Jacobi Numbers

Time limit: 1s

Today, a new paper has been published in the *Bulletin of Apocryphal Pioneers in Computation*. According to this paper, the forgotten German number theorist Wahnfried Imaginus Jacobi (1806–1853), while still a secondary student in Potsdam, investigated the decomposition of integers into sums of cubes. Among the examples noted in the surviving fragments of his notebooks are

$$2025 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + 7^3 + 8^3 + 9^3$$

and the more curious expression

$$3 = 1^3 + 1^3 + 1^3 = 4^3 + 4^3 + (-5)^3\,,$$

Carl Gustav Jacob Jacobi (1804–1851), famous brother of Wahnfried Imaginus. Public domain on Wikimedia Commons

which shows that a solution need not be unique. Jacobi restricted his attention to small integers and probably did not know the decomposition

$$3 = 569\,936\,821\,221\,962\,380\,720^3 + (-569\,936\,821\,113\,563\,493\,509)^3 + (-472\,715\,493\,453\,327\,032)^3\,,$$

which was discovered only recently.[4] However, Jacobi did manage to prove that a decomposition into cubes always exists for all positive integers up to 9241, the 28th cuban prime of the first kind. Although his work was never published, references to the method appear in a marginal annotation in an 1823 letter to his famous brother Carl Gustav Jacob.

Given a positive integer $n$, output a list of at most 10 000 integers between $-10\,000$ and 10 000 such that the sum of their cubes equals $n$.

## Input

The input consists of:

- One line with an integer $n$ ($1 \leq n \leq 9241$), the number to decompose into cubes.

## Output

Output an integer $k$ ($1 \leq k \leq 10\,000$), the number of terms in your solution, followed by $k$ integers $a_1, \ldots, a_k$ ($-10\,000 \leq a_i \leq 10\,000$ for each $i$), such that $a_1^3 + \cdots + a_k^3 = n$.

If there are multiple valid solutions, you may output any one of them.

| Sample Input 1 | Sample Output 1 |
|---|---|
| 2025 | 9<br>1 2 3 4 5 6 7 8 9 |

[4] Booker, Andrew R.; Sutherland, Andrew V. (2021), "On a question of Mordell", *Proceedings of the National Academy of Sciences*, **118** (11)

| **Sample Input 2** | **Sample Output 2** |
|---|---|
| 45 | 3<br>2025 -2369 1709 |

| **Sample Input 3** | **Sample Output 3** |
|---|---|
| 15 | 3<br>-1 2 2 |

| **Sample Input 4** | **Sample Output 4** |
|---|---|
| 9241 | 2<br>-55 56 |

# K Knowing the Clock

Time limit: 1s



It is the 25th of November, 1625. You have put a lot of time into making a mechanical watch, but you are unsure if the hands are placed correctly. Even a watch that is not running is correct twice a day, but if the hands of the watch do not correspond to a real time, it is never correct at all. Without wasting any more time, you measure the angles of the hands and check whether they correspond to a real time.

For example, consider the first sample input, visualized in Figure K.1: if the minute hand points to a quarter past (90 degrees from 12 o'clock), then the hour hand cannot point exactly to 2 o'clock (60 degrees from 12 o'clock).

You know that both hands of the watch are moving continuously.

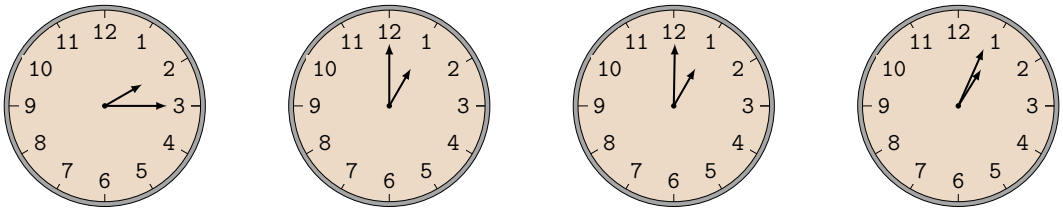


Figure K.1: Illustrations of the sample inputs. The first and third sample inputs *do not* correspond to a real time. The second and fourth sample inputs *do* correspond to a real time.

## Input

The input consists of:

- One line with two integers $h$ and $m$ ($0 \leq h, m \leq 359$), the exact clockwise angle that the hour hand is from 12 o'clock and the exact clockwise angle that the minute hand is from 12 o'clock. Both angles are given in degrees.

## Output

If the angles of the hands correspond to a real time, output "`yes`". Otherwise, output "`no`".

| Sample Input 1 | Sample Output 1 |
|---|---|
| `60 90` | `no` |

| Sample Input 2 | Sample Output 2 |
|---|---|
| `30 0` | `yes` |

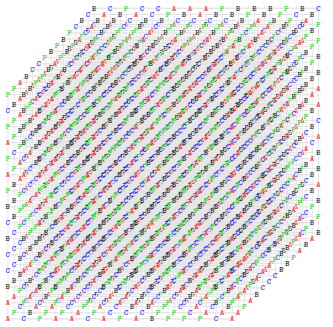
| **Sample Input 3** | **Sample Output 3** |
|---|---|
| 30 1 | no |

| **Sample Input 4** | **Sample Output 4** |
|---|---|
| 32 24 | yes |

# L Linguistic Labyrinth

Time limit: 15s

It is the 25th of December, 2025. As a Christmas tradition, you gather a group of friends to solve a puzzle. Among your friends are *wordcels* and *shape rotators*, who are respectively better at thinking with words and with mental images. This puzzle challenges even the smartest wordcel and the most brilliant shape rotator:



Spoiler alert: this is one of the secret test cases.

There is a 3-dimensional grid with points at all integer coordinates $(x, y, z)$ with $1 \leq x, y, z \leq n$, and each point has a label associated with it, which is either '`B`', '`A`', '`P`', or '`C`'. In this grid, you need to find occurrences of the curly word "`BAPC`". A curly word "`BAPC`" is a collection of four points in the grid such that:

- The labels spell out "`BAPC`" (in this order).
- The angle that the triplet "`BAP`" makes is 90 degrees: the vectors from `B` $\rightarrow$ `A` and from `A` $\rightarrow$ `P` form a 90-degree angle.
- The angle that the triplet "`APC`" makes is 90 degrees: the vectors from `A` $\rightarrow$ `P` and from `P` $\rightarrow$ `C` form a 90-degree angle.

Note that the two angles do not need to be axis-aligned. As an example, see the third sample case, visualized in Figure L.1.

How many occurrences of the curly word "`BAPC`" are in the given grid?

## Input

The input consists of:

- One line with an integer $n$ ($1 \leq n \leq 22$), the size of the grid.
- $n$ blocks of $n + 1$ lines. Each block of $n + 1$ lines consists of:
  - One line with a hyphen (`-`), to make the input more human-readable.
  - $n$ lines with $n$ characters, each character being either '`B`', '`A`', '`P`', or '`C`', representing all labels of one horizontal layer of the 3-dimensional grid.

## Output

Output the number of curly words "`BAPC`" in the 3-dimensional grid.

| Sample Input 1 | Sample Output 1 |
|---|---|
| 1<br>-<br>B | 0 |

**Sample Input 2**

```
2
-
PA
PB
-
CC
PB
```

**Sample Output 2**

```
2
```

**Sample Input 3**

```
3
-
BBB
BCB
BCB
-
BBC
CBA
BBB
-
BBB
BPB
BBB
```

**Sample Output 3**

```
2
```

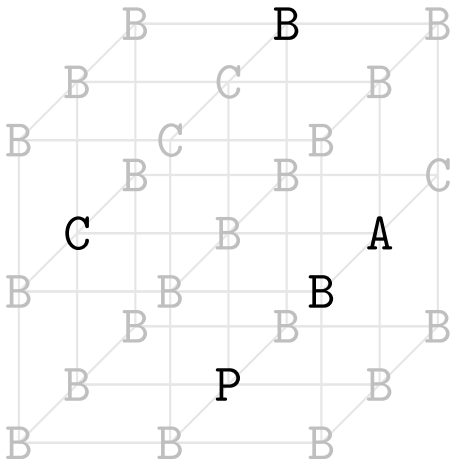


Figure L.1: Visualization of the third sample input. In this grid, there are two curly words "BAPC", using the highlighted letters.