Nordic Collegiate Programming Contest

NCPC 2025

October 4, 2025



Problems

A Arithmetic Adaptation
B Bohemian Bookshelf
C Crochet Competition
D Dune Dash
E Egyptian Equality
F Follower Forensics
G Gotta Trade Some of 'Em
H Hidden Permutation
I Instagraph
J Jump
K km/h

Do not open before the contest has started.

Advice, hints, and general information

- The problems are **not** sorted by difficulty.
- Your solution programs must read input from *standard input* (e.g. `System.in` in Java or `cin` in C++) and write output to *standard output* (e.g. `System.out` in Java or `cout` in C++). For further details and examples, please refer to the documentation in the help pages for your favorite language on Kattis.
- For information about which compiler flags and versions are used, please refer to the documentation in the help pages for your favorite language on Kattis.
- Your submissions will be run multiple times, on several different inputs. If your submission is incorrect, the error message you get will be the error exhibited on the first input on which you failed. E.g., if your instance is prone to crash but also incorrect, your submission may be judged as either "Wrong Answer" or "Run Time Error", depending on which is discovered first. The inputs for a problem will always be tested in the same order.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is "No comment, read problem statement", indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem, or that the answer to the question is simply irrelevant to solving the problem.
- In general we are lenient with small formatting errors in the output, in particular whitespace errors within reason, and upper/lower case errors are often (but not always) ignored. But not printing any spaces at all (e.g. missing the space in the string "`1 2`" so that it becomes "`12`") is typically not accepted. The safest way to get accepted is to follow the output format exactly.
- For problems with floating point output, we only require that your output is correct up to some error tolerance. For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that
  - If the correct answer is 0.05, any answer between 0.0499 and $.0501$ will be accepted.
  - If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.

  Any reasonable format for floating point numbers is acceptable. For instance, "17.000000", "0.17e2", and "17" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Problem A
Arithmetic Adaptation

Practice does make perfect! You have finally achieved proficiency at the task of adding two small nonzero integers a and b to compute their sum $a+b$. Before you move on to studying four-digit numbers, you want to achieve mastery by also understanding the inverse problem: given integer s, determine nonzero a and b such that $a + b = s$. None of the numbers may use more than 3 digits.



Postcard showing a Norwegian by solving Sample 3. Coloured black and white photo, National Library of Norway.

Input

The input consists of:

- One line with an integer s such that $-999 \leq s \leq 999$.

Output

Output two integers a (with $-999 \leq a \leq 999$ and $a \neq 0$) and b (with $-999 \leq b \leq 999$ and $b \neq 0$) such that $a + b = s$. If there is more than one valid solution, you may output any one of them.

Explanation of Sample Input 1

On input "`10`", the output "`3 7`" is correct because $3 + 7 = 10$. Note that many other outputs would also be correct, such as "`2 8`", "`11 -1`", or even "`-849 859`". On the other hand, the answer "`4 7`" would be *wrong* (because $4 + 7 \neq 10$), and so would "`10 0`" (because both a and b must be nonzero) and "`1000 -990`" (because both a and b must have at most three digits.)

Sample Input 1	Sample Output 1
`10`	`3 7`

Sample Input 2	Sample Output 2
`-1`	`-2 1`

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

Sample Input 4	Sample Output 4
`0`	`-999 999`

This page is intentionally left blank.

Problem B
Bohemian Bookshelf

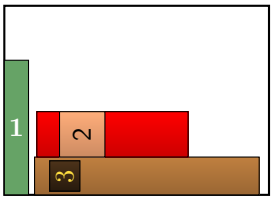


Image of antiquarian books on a shelf from the Dr. Philippe Garigue Collection. Credit: Taylor Tryburski, CC BY-SA 4.0.

When it comes to shelving books, bibliophiles and interior decorators rarely agree: vertical or horizontal—what's the right way? Ever skeptical of orthodoxy, you embrace both! Your bookshelf displays a composed mix: upright volumes standing with quiet discipline next to a single, neatly arranged stack, like a literary ziggurat. The effect signals bohemian chic with a touch of endearing academic absent-mindedness and just enough polish to suggest you've definitely read some of the books.

A book is characterised by its spine height and thickness. It can fit upright if its height does not exceed the bookshelf's height. Alternatively, books can form a stack by being laid flat on their sides on top of each other; for aesthetic reasons the books forming such a stack must be arranged in nonincreasing order of spine height. Their total thickness may not exceed the height of the bookshelf. The total thickness of the upright books and the width of the stack may not exceed the width of the bookshelf.

The books in Sample 1 can be arranged like this:



Input

The input consists of:

- One line with integers N, H, W ($2 \leq N \leq 100$, $130 \leq H \leq 350$, $300 \leq W \leq 900$), the number of books you want to display, the height of the bookshelf in millimeters, and the width of the bookshelf in millimeters,
- N lines, one for each book, with two integers h, t ($76 \leq h \leq 483$, $5 \leq t \leq 60$), the book's spine height and thickness in millimeters. On its own, every book fits on the shelf either standing up or on its side, so we also have $h \leq \max(W, H)$.

Output

Output two lines.

The first line starts with the word "`upright`", followed by the indices of the upright books. The second line starts with the word "`stacked`", followed by the indices of the books in the stack, in order from bottom to top. There must be at least one upright book and at least one stacked book. Books are indexed $1, \ldots, N$.

If there is more than one solution, you may output any one of them. If no such arrangement is possible, print "`impossible`".

Sample Input 1	Sample Output 1
3 250 350 178 32 200 60 297 50	upright 1 stacked 3 2

Sample Input 2	Sample Output 2
2 300 300 290 60 290 60	impossible

Problem C
Crochet Competition



Crocheting. License: CC BY-SA 3.0 by floret on Wikimedia Commons.

The National Crochet Pattern Competition (NCPC) is an annual team-based event for crocheters of all skill levels, from beginner to expert. Teams design, write, and crochet original patterns within a time limit, which can range from minutes to a full week depending on the complexity—sometimes spanning weekends. Contestants may take breaks to eat, sleep, or go home, but the timer continues running.

NCPC has asked you to develop a system that calculates the total competition duration from the start and end times, provided with weekday, hour, and minute. The reported duration should be precise, correctly expressed in days, hours, and minutes, using as few components as necessary while minimizing the sum of the integer values.

Input

Input consists of two lines.

- The first line gives the competition's start time.
- The second line gives the competition's end time.

Times are formatted as "`Www HH:MM`", where

- `Www` is the three-letter weekday abbreviation: `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat`, or `Sun`.
- `HH` the two-digit hour, from 00 to 23.
- `MM` is the two-digits minutes, from 00 to 59.

If the start and end times are identical, the competition lasts exactly one week.

Output

Output the competition duration in days, hours, and minutes, using the singular or plural form as appropriate (e.g., 1 day, 2 days). List the components in order – days, hours, minutes – separating three components with commas and two with "and"; omit any component with a value of zero.

Sample Input 1	Sample Output 1
`Mon 08:00` `Mon 15:00`	`7 hours`

Sample Input 2	Sample Output 2
`Mon 10:00` `Wed 08:59`	`1 day, 22 hours, 59 minutes`

Sample Input 3	Sample Output 3
Fri 20:00 Mon 08:00	2 days and 12 hours

Problem D
Dune Dash

You signed up for the *Dune Dash*, a running race across the desert. Everything went well — except that in the excitement, you forgot to start *StrideTrack*, the app that records how far you've run. All you have now are the official checkpoint locations, but not the order in which you passed through them.



A runner during a 15-kilometer run through the Grand Bara Desert in Djibouti. Public domain photo by Joshua Bruns.

Formally, the race consisted of N checkpoints, each given by its coordinates in the Euclidean plane. The sequence in which they were visited is unknown to you, but the organizers designed the course to prevent anyone from straying off route. In particular, if $q_1, q_2, \ldots, q_N$ is the correctly ordered list of the checkpoints along the race, then for every triple $i < j < k$ it holds that

$$\text{dist}(q_i, q_k) > \max(\text{dist}(q_i, q_j), \text{dist}(q_j, q_k))$$

where $\text{dist}(p, q)$ denotes the Euclidean distance between points p and q. Your task is to determine the total length of the race.

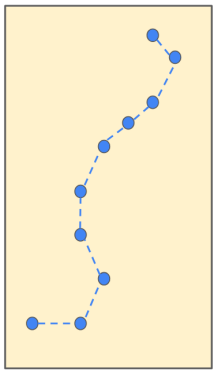


Figure D.1: Illustration of sample 2. The dashed line shows where the race went.

Input

The first line consists of the integer N ($2 \leq N \leq 2 \cdot 10^5$). The following N lines each contain two integers x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$). These are the coordinates of each checkpoint.

The checkpoints are not necessarily in the order in which they were visited during the race. It is guaranteed that there is some ordering of the checkpoints such that they satisfy the distance requirements above.

The N points given in the input are all distinct.

Output

Print one floating point number, the length of the race. Your answer will be correct if it has an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
3
1 0
0 0
1 1
```

Sample Output 1

```
2.0
```

Sample Input 2

```
10
-1 -7
-1 -11
0 -9
2 2
1 -2
2 -1
3 1
-1 -5
0 -3
-3 -11
```

Sample Output 2

```
17.186912597118443
```

Problem E
Egyptian Equality



Khafre's pyramid at Giza. At the top of the pyramid you can see many remaining casing stones. Image by Daaaveee, commons.wikimedia.org

Friendly but competing archaeologists Ameena and Bilal are searching for the lost entrance to the *Nomarch's Celestial Pyramid of Cauris*. Over the Millennia, much of the outer layer has been stripped away, so the entrance must lie behind one of the remaining "casing" stones – the smooth limestone blocks that once covered the entire monument – on the north face of the pyramid.

To avoid interference between their sensor arrays, Ameena and Bilal will divide the north face into two connected regions and scan one region each. For fairness, each region must contain the same number of casing stones.

They model the pyramid's north face as a triangular grid. The grid has N rows and $2N - 1$ columns, with the pyramid occupying a subset of these cells: the middle cell of the first row, the middle three cells of the second row, the middle five of the third row, and so on. A nonempty subset of the cells in the pyramid are casing stones.

Your task is to partition the pyramid into two regions A and B such that:

- Every cell in the pyramid belongs to exactly one region.
- Each region is connected. (A region is connected if it is possible to walk between any two of its cells by moving between adjacent cells. Two cells are adjacent if they share a side.)
- The two regions contain the same number of casing stones.

Input

The first line consists of the integer N ($2 \leq N \leq 300$). The following N lines each contain a string of $2N - 1$ characters. These strings describe where the casing stones are. The character `#` represents a cell that does not belong to the pyramid, `.` is a cell that is not a casing stone, and `C` is a casing stone. There will be at least one casing stone in the pyramid.

Output

If it is impossible to partition the pyramid in a valid way, print `impossible`. Otherwise, print N lines where each line contains a string of length $2N - 1$. These lines describe the partition of the pyramid. The character `#` represents a cell that does not belong to the pyramid, `A` represents a cell that belongs to Ameena, and `B` represents a cell that belongs to Bilal. If there are multiple valid solutions, you can print any one of them.

Sample Input 1	Sample Output 1
4 ###.### ##C..## #..CC.# .C.....	###B### ##BBB## #AABAA# AAAAAAA

Sample Input 2	Sample Output 2
2 #.# .C.	impossible

Problem F
Follower Forensics

Your social media company ConnectHub prides itself on how easily information spreads through its network.

Accounts can share posts directly with their followers using the share button, or with the accounts they follow by posting in the comments. ConnectHub is famous for the virality of its users: a post from any account was able to reach every other account, possibly through a chain of shares and comments.

But disaster struck: a database mishap wiped out all records of who follows whom. One piece of data may have survived, though — for each account, you have the number of its followers and the number of accounts it follows. Maybe even those numbers have been corrupted, but it's all you have.

Your task is to rebuild the missing database so that these counts are correct, and ConnectHub once again has full virality.

Input

The input consists of:

- One line with an integer n, the number of accounts, where $1 \leq n \leq 100\,000$.
- One line with n integers $a_1, \ldots, a_n$ satisfying $0 \leq a_i \leq n$ for each i, where a_i is the number of accounts followed by the ith account.
- One line with n integers $b_1, \ldots, b_n$ satisfying $0 \leq b_i \leq n$ for each i, where b_i is the number of accounts following the ith account.

Additionally, it is guaranteed that

$$\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i \leq 2\,000\,000.$$

Note that the input is not guaranteed to allow for any reconstruction, fully viral or not. For those familiar with that terminology, full virality is equivalent to weak connectivity.

Output

If a reconstruction is possible, output a network

- One line with two integers: the number n of accounts, and the number m of connections. This is the same n as in the input.
- Then m lines, each consisting of two integers a and b, specifying that account a follows account b.

Otherwise output "`impossible`".

Note that it is not allowed for an account to follow the same account twice, nor to follow itself.

Sample Input 1

```
5
2 1 2 1 3
0 2 2 3 2
```

Sample Output 1

```
5 9
2 4
4 5
3 5
3 2
5 2
1 3
1 4
5 3
5 4
```

Sample Input 2

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

Sample Output 2

```
impossible
```

Problem G
Gotta Trade Some of 'Em



Two Game Boy systems connected with a DMG-04 (link cable). Public domain photo by KoS.

The kids at school are obsessed with a new video game of collectible pocket monsters called *Pokémon*. Their goal is to "catch 'em all" by completing the Pokédex, meaning they want to catch a copy of every Pokémon. Typically they do this by playing the game, encountering Pokémon, and catching them with a Pokéball.

Pokémon games are released in different variants. Each variant contains a number of Pokémon, some of which are exclusive to that variant. For example, only *Pokémon Blue* (variant 1) allows encounters with the Pokémon known as Meowth. Thus, a kid owning *Pokémon Red* (variant 2) cannot catch Meowth, and therefore must trade with a friend who has a Meowth to fill the Pokédex. This friend may either own Pokémon Blue or have received Meowth through another trade.

Your task is to distribute Pokémon variants to the kids so that, through trading among friends, every kid can eventually collect at least one of each Pokémon. Each kid receives exactly one Pokémon game variant. Each variant comes with enough copies of its Pokémon, both its exclusives and the non-exclusives, ensuring trades are always possible.

Input

The input consists of:

- One line with three integers n, m, and k: the number n of kids, with $1 \leq n \leq 100\,000$, the number m of friendships, with $1 \leq m \leq 200\,000$, and the number k of different game variants, with $1 \leq k \leq 100\,000$.
- m lines, each containing two integers a and b with $1 \leq a < b \leq n$, representing that a and b are friends.

Output

Ouput n integers, one for each kid, representing which game variant that kid should get. Your output will be considered correct if all kids can fill their Pokédex by using any number of trades with their friends.

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

If there is no assignment of game variants to kids that allows all kids to fill their Pokédex, output `impossible`.

Sample Input 1	Sample Output 1
8 5 2 1 2 2 5 3 4 5 6 7 8	1 2 1 2 1 2 1 2

Sample Input 2

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

Sample Output 2

```
impossible
```

Sample Input 3

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

Sample Output 3

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

Problem H
Hidden Permutation

When a fixed shuffle is applied to a deck of cards repeatedly, the deck will eventually return to its original order. The number of times the shuffle must be applied before this happens is called the *period* of the deck for that shuffle. In the present task, we consider the inverse question: rather than starting with a shuffle and finding its period, you are told, for each possible period, how many decks of cards have that period. Your task is to construct any shuffle consistent with this information.

Formally, there is a hidden permutation $f = (f_1, f_2, \ldots, f_N)$, where N is a given positive integer. A permutation is a list of numbers where each number from 1 to N occurs exactly once. Let $x = x_1 x_2 \ldots x_N$ be a binary string. We say that $f(x)$ (f applied to x) is the new binary string $x_{f_1} x_{f_2} \ldots x_{f_N}$, and the *period* of x is the smallest positive integer p such that

$$\underbrace{f(f(\cdots f(x)\cdots))}_{p \text{ times}} = x.$$

It can be proven that for any permutation f and any binary string x of the same length, the period of x exists. In other words, if you apply f to x enough times, you eventually get back x.

You are given the number N, and for each possible period p you are given the number of binary strings (out of all 2^N possible binary strings) that have period p. Your task is to find any permutation f that corresponds to the given information, or conclude that there is no such permutation.

Input

The input consists of:

- One line with integers N, K ($2 \leq N \leq 100$, $1 \leq K \leq 1000$), the length of the permutation and the number of possible periods.
- One line with K integers $p_1, p_2, \ldots, p_K$ ($1 \leq p_i \leq 10^9$). These are all the possible periods that binary strings of length N can have, with respect to the hidden permutation. The numbers p_i are all distinct, and are sorted in increasing order.
- One line with K integers $m_1, m_2, \ldots, m_K$ ($1 \leq m_i \leq 2^{100}$). This means that there are m_i binary strings with period p_i.

Note that the numbers m_i can be quite large!

Output

If there is no permutation f that corresponds to the given information, print `impossible`. Otherwise, print one line with N integers $f_1, f_2, \ldots, f_N$, the permutation that you chose. Recall that the permutation should contain every integer from 1 to N exactly once. If there are multiple solutions, you can print any one of them.

Sample Explanation

In the second sample, the input almost corresponds to the permutation $(2, 3, 4, \ldots, 91, 1)$. The only error is that the last digit in the last number of the input should be 0 instead of 1.

In the third sample, all 4 binary strings have period 1, which means that the identity permutation $(1, 2)$ is a valid answer.

Sample Input 1

```
7 6
1 2 3 4 6 12
4 4 12 24 12 72
```

Sample Output 1

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

Sample Input 2

```
91 4
1 7 13 91
2 126 8190 247588007857076054979824013 1
```

Sample Output 2

```
impossible
```

Sample Input 3

```
2 1
1
4
```

Sample Output 3

```
1 2
```

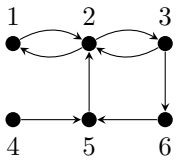
Problem I
Instagraph

A celebrity in a social network is somebody with many followers, but who doesn't follow them back. More precisely, a person is a *celebrity* for a group of people, if

- every member of the group follows the person,
- the person follows nobody in the group.

The *celebrity centrality* of person v, written $\mathrm{CC}(v)$, is the maximum size of such a group.

We model the social network as a directed graph with N vertices $1, \ldots, N$. A directed edge from u to v means that person u follows person v. For example, in



we have $\mathrm{CC}(1) = 0$, $\mathrm{CC}(2) = 1$, and $\mathrm{CC}(5) = 2$.

Your task is to find a vertex v with the maximum celebrity centrality $\mathrm{CC}(v)$. In case of a tie, choose the smallest v.

Input

The input consists of

- One line with two integers N and M ($1 \leq N \leq 100\,000$, $0 \leq M \leq 1\,000\,000$), the number of vertices and the number of directed edges.
- M lines with two distinct integers u and v ($1 \leq u, v \leq N$), indicating a directed edge from u to v. There are no duplicate edges.

Output

Output two integers: the smallest v with the maximum celebrity centrality and the value $\mathrm{CC}(v)$.

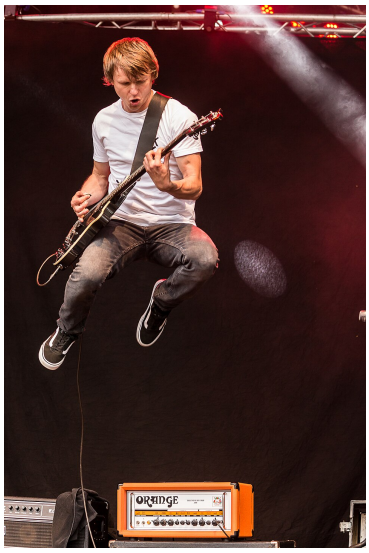
Sample Input 1	Sample Output 1
6 8 1 2 2 1 2 3 3 2 3 6 4 5 5 2 6 5	5 2

Sample Input 2	Sample Output 2
1 0	1 0

Problem J
Jump

The Unsightly Halfwits are putting on a wild concert in their natural habitat of Húsavík. Their lead singer, Sveinn "Springheel" Jónsson, is leaping about the stage with vigorous energy, turning every note into a spectacle. But keeping in rhythm while bouncing so chaotically is no easy task, and you can't help but wonder how long he can keep this up.

Sveinn starts with an energy level of E. The song plays a beat exactly every N milliseconds. His first jump launches at time 0, perfectly in sync with the music. As long as his energy E is strictly positive ($E > 0$), each jump lasts exactly E milliseconds. When his energy runs out with $E = 0$, he collapses and the show is over. If he still has energy after landing, he immediately springs up again for the next jump. When Sveinn lands exactly on a beat, he gains $+1$ energy from sheer enjoyment. But if he's even slightly off-beat, his energy drops by 1.



Ballsout at Metal Frenzy Open Air 2022, image by S. Bollmann, commons.wikimedia.org.

Input

The input consists of:

- One line with two integers N, E, the number of milliseconds between beats and the initial energy level of the lead singer. These will satisfy $1 \leq N \leq 3 \cdot 10^5$ and $1 \leq E \leq 10^9$.

Output

Output a single integer, the number of milliseconds that will pass before Sveinn stops jumping. If he never stops, instead print `infinity`.

Sample Input 1	Sample Output 1
`7 9`	`59`

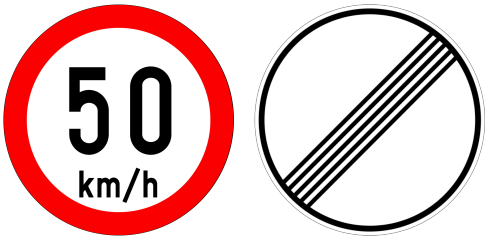
Sample Input 2	Sample Output 2
`7 4`	`infinity`

Sample Input 3	Sample Output 3
`100001 123123123`	`7581167638793024`

This page is intentionally left blank.

Problem K
km/h

Many countries have a "national speed limit": a default maximum speed that applies unless a *speed limit sign* explicitly sets a lower one. Such a speed limit sign shows a number and overrides the default while it is in force. By contrast, a *derestriction sign* shows no number at all; it cancels the previous restriction and restores the national speed limit, which is always strictly higher than any speed limit sign.



A speed limit sign (left) and a derestriction sign (right). National variants differ, but are regulated by the *Vienna Convention on Road Traffic* (1968).

You pride yourself on being a careful and attentive motorist. Whenever you encounter a speed limit sign, you smoothly and safely adjust your vehicle's speed. Alas, despite your otherwise impeccable memory, the derestriction sign always leaves you guessing. You simply cannot remember the national speed limit – only that it was a neat multiple of ten.

As you encounter each sign – limit or derestriction – determine the fastest you can go while abiding the law, based only on what you've seen so far.

Input

The input consists of:

- One line with an integer N with $1 \leq N \leq 100$, the number of signs.
- N lines, one for each sign. Each line describes either a speed limit or a derestriction. A speed limit is an integer L with $10 \leq L \leq 160$, the speed limit that applies from that sign. A derestriction is the forward slash"/", reminiscent of the sign's diagonal bar. The first line contains a speed limit. There are never two consecutive derestrictions.

Output

Output N integers, the maximum legal speed that can be inferred when passing the ith sign for $1 \leq i \leq N$.

Sample Input 1	Sample Output 1
8 90 85 / 119 / 85 85 /	90 85 100 119 120 85 85 120

This page is intentionally left blank.